

Structural Analysis of Seven Hash Functions

Submitted to the NGCC

Yufei Yuan^{1,2,*}, Ruichen Wu^{1,2,*}, Shanpeng Wei^{1,2,*}, Junxu Shen^{1,2,*},
Jinpeng Liu^{1,2,*}, and Yixin Zhang^{1,2,*}

¹ Trusted Computing and Information Assurance Laboratory, Institute of Software
Chinese Academy of Sciences, Beijing, 100190, China

² University of Chinese Academy of Sciences, Beijing, 100049, China
(Contact: yufei2021@iscas.ac.cn)

Abstract. The Institute of Commercial Cryptography Standards (ICCS) launched the Next-generation Commercial Cryptographic Algorithms Program (NGCC) and invited worldwide comments on draft submission requirements and evaluation criteria for cryptographic hash algorithms. We analyze seven hash functions submitted to the NGCC and establish the following results:

1. Message differences that cancel in every key injection give explicit collisions for all four variants of **MoFang** with fixed output lengths and both XOF variants at every finite output length.
2. Two distinct states of **Neulaser** become equal after one update, giving collisions for all three variants with the initialization used by the v2 reference and optimized implementations.
3. An invariant subspace of **CHIME-512** gives an upper bound of 2^{64} hash evaluations for finding a collision when shifts act separately on 64-bit words, as in both submitted implementations.
4. **CHAMP**'s determinant constraint gives upper bounds of 2^{192} and 2^{384} hash evaluations for finding collisions in its 512- and 1024-bit variants, respectively.
5. We exhibit explicit free-start collisions for all three **QSH** variants. In the semi-free-start setting, both messages and every tree node share one chosen complete reset. We prove upper bounds of 2^{64} , 2^{128} , 2^{128} compression evaluations for finding collisions with 512-, 768-, and 1024-bit digests, respectively, plus constant preparation and finalization costs.
6. In both versions of **WChain**, the message expansion preserves invariant sets and admits sequences of expanded blocks with periods one, three, and six over all prescribed steps.
7. On a set of 256 message blocks, **Cuishen**'s message expansion acts as a cyclic shift of eight binary coordinates throughout all 64 rounds. For two blocks from this set, evaluated with the same initial chaining register and counter, the XOR differences between round keys form a sequence whose period divides eight.

* All authors contributed equally to this work.

The stated searches for **CHIME-512**, **CHAMP**, and **QSH** have success probabilities greater than 0.39, without assuming uniform outputs; their exponential budgets are theoretical.

Keywords: Hash functions · Cryptanalysis · Collision attacks · Invariant subspaces · Message expansion

1 Introduction

The Institute of Commercial Cryptography Standards (ICCS) launched the Next-generation Commercial Cryptographic Algorithms Program (NGCC), a global initiative to solicit new commercial cryptographic algorithms [7]. The public evaluation of these submissions calls for independent cryptanalysis of their constructions. We analyze seven submitted hash functions. The definitions used in the analysis are stated in the relevant sections and appendices, with citations to the submission documents. Our results concern message expansion, round functions, and the way these components enter the hash construction.

We first present explicit collisions and bounds on the cost of finding collisions at the stated initialization interfaces. QSH admits explicit free-start collisions and a semi-free-start bound in the tree mode derived from exact core classifications. We then examine the message expansions of WChain and Cuishen. Table 1 summarizes the results in this order.

Commuting transformations and equality spaces are established tools for studying invariant structure [11, Secs. 3.1–3.2]. For QSH, we classify images of spaces defined solely by equalities between words and relations induced by permutations of word positions. We trace their consequences through modular message injection, CV projection, and legal tree processing. Frameworks for tree hashing make the distinction between an internal primitive and its hash interface explicit [1, Sec. 3]. Our proofs specify the resets, flags, padding, and transmitted CVs needed for each consequence.

1.1 Contributions

Our results are as follows:

1. Four independent mask bits define message differences in **MoFang** whose contributions from the two message branches cancel at every key injection. The cancellation holds for any incoming chaining value, giving at least sixteen equivalent choices per complete message block. Independent choices across blocks yield collisions for all four variants with fixed output lengths. We also prove that the same message pairs collide for both XOF variants at every finite output length, under either the documented or the implemented squeezing transitions.
2. We exhibit two distinct internal states of **Neulaser** that become equal after one update. The differing word affects the next state only through its local feedback, which agrees for the two inputs. Direct big-endian $V \parallel$

K initialization, as used by the v2 submission’s reference and optimized implementations, maps explicit equal-length messages to these states. This gives collisions for all three variants with their prescribed IVs, padding, and all rounds.

3. We construct 2^{240} distinct 126-byte messages whose **CHIME-512** evaluations remain in an invariant subspace when shifts act separately on 64-bit words, as in both submitted implementations. The subspace is preserved through both absorption steps and all 15 rounds per block, restricting the digests to at most 2^{128} values.³ Uniform sampling from this family gives a collision search using at most 2^{64} hash evaluations. The bound holds with both specified and implemented byte orders.
4. The generators of **CHAMP** have equal determinants, which confines all products for a fixed message length to at most $p(p^2 - 1)$ matrices over the underlying field $\mathbb{F}_p$. The final encoding preserves this bound. Sampling messages of the same length as the digest gives collision searches using at most 2^{192} and 2^{384} hash evaluations for the 512- and 1024-bit variants, respectively.
5. We construct explicit free-start collisions for **QSH** using the invertibility of its compression function with respect to the incoming state. For the semi-free-start setting, the exact classifications of equality spaces and coordinate relations give a bound on the number of complete CVs. An invertible prefix embeds the corresponding message family into legal padded messages, and equal leaf CVs survive the specified tree processing.
6. Restricting each 64-bit message word to zero or $2^{64} - 1$ reduces **WChain**’s message expansion to one 18-bit permutation in Version I and two independent copies in Version II. We enumerate the reduced permutation’s cycles and construct sequences of expanded blocks with periods one, three, and six. These sequences include all expanded blocks used in the 18 or 24 data rounds and the block used for final whitening. We give explicit padded messages with periodic expansions and show that the checksum preserves membership in the invariant set.⁴
7. We identify a 256-element, eight-dimensional invariant subspace of **Cuishen**’s message register on which expansion becomes a cyclic shift of eight binary coordinates. The induced recurrence has least periods one, two, four, and eight, and these periodic patterns persist throughout all 64 prescribed rounds. For two values of the message register from this subspace and the same initial chaining register and counter, the XOR differences between round keys form a sequence whose period divides eight. All 256 values can be supplied as the first 1024-bit message block under the prescribed IV.⁵

³ The designers have addressed this issue in the CHIME erratum by revising the round constants to use distinct 64-bit words.

⁴ Our analysis of the message expansion does not establish a reduction in the claimed security of the hash function.

⁵ Our analysis of the recurrence and differences between round keys does not establish a reduction in the claimed security of the hash function.

Table 1. Results and initialization conditions. Search bounds have success probability greater than 0.39, with at most one record per sample. CHIME and CHAMP count hash evaluations; QSH counts compressions with separate preparation and finalization costs.

Algorithm	Result and conditions
MoFang	Explicit collisions for all four variants with fixed output lengths and both XOFs at every finite output length, under either squeezing convention in Section 3.3; at least sixteen equivalent choices per complete message block.
Neulaser	Explicit collisions for all three variants under direct big-endian $V \parallel K$ initialization, as used by both v2 implementations.
CHIME-512	A collision search using at most 2^{64} evaluations under shifts on individual 64-bit words, as in both submitted implementations; the selected messages have at most 2^{128} outputs.
CHAMP	Collision searches using at most 2^{192} and 2^{384} evaluations for 512- and 1024-bit digests.
QSH	Explicit free-start collisions with separately chosen complete resets. Semi-free-start searches in the tree mode use $2^{64}, 2^{128}, 2^{128}$ compression evaluations, plus 15 evaluations of the inverse permutation and three finalization compressions, with one chosen 64-word reset shared by both messages and all nodes.
WChain	Invariant sets in both message expansions and explicit sequences of expanded blocks with periods one, three, and six, including the block used for final whitening.
Cuishen	A cyclic shift of eight binary coordinates on 256 message register values over 64 rounds; periodic XOR differences between round keys for fixed initial chaining and counter registers.

The **CHIME-512**, **CHAMP**, and semi-free-start **QSH** searches give theoretical upper bounds, each with success probability greater than 0.39 and at most one search record per sampled input. CHIME and CHAMP count hash evaluations; QSH counts compression evaluations, with preparation and finalization stated separately. The bounds apply to arbitrary output distributions.

2 Preliminaries

2.1 Notation and targets

We write $\{0, 1\}^n$ for n -bit strings, $\oplus$ for bitwise exclusive or, and $\parallel$ for concatenation. Byte offsets start at zero. Hexadecimal byte strings are written in reading order. Words use big-endian encoding unless an algorithm section states otherwise. The functions $\text{rotl}_w(x, t)$ and $\text{rotr}_w(x, t)$ rotate a w -bit word by t positions. A complement is taken at the operand’s stated width. The expression $f \circ g$ applies g first. The strings 0^n and 1^n have n bits; repetitions of hexadecimal byte strings are stated in bytes. Hexadecimal values are printed in monospaced type, and other numeric constants are decimal. A w -bit word is identified with its canonical

integer in $[0, 2^w - 1]$. Integer constants in word operations are encoded at that width; any modular reduction is stated explicitly.

A collision for a hash function H is a pair of distinct unpadded messages $M \neq N$ such that $H(M) = H(N)$, with both evaluations using the prescribed padding and finalization. A collision in a component f consists of distinct inputs with equal outputs. To obtain a hash collision from an equality of internal states, we construct messages that produce those states and show that equality is preserved through the remaining rounds, feed-forward, counters, padding, and finalization.

Unless a result explicitly parameterizes initialization, H includes its prescribed initialization. For a compression map $f(I, M)$, a *free-start collision* consists of distinct input pairs satisfying $f(I, M) = f(I', M')$, with both initial states selectable. A *semi-free-start collision* additionally requires $I = I'$ and $M \neq M'$ [10, Sec. 1]. For QSH we extend these settings to the complete tree mode: QSH_I uses one complete 64-word reset at every node. The free-start setting permits separate resets for the two messages; the semi-free-start setting fixes one chosen reset shared by both messages and all nodes. A node's complete transmitted chaining value (CV), the full compression state, and the final digest are distinct objects; their widths are stated at each relevant interface.

A permutation distinguisher receives oracle access to a permutation and chooses its input state. Its advantage is the difference between acceptance probabilities for the specified permutation and a uniformly random permutation of the same state space. For a public permutation, we account for both oracle queries and local computation. Applying a component test through the hash interface requires both legal messages that reach the queried states and a predicate observable after the intervening mode operations and digest extraction.

An invariant set for message expansion is preserved by each expansion step. Round keys may also depend on chaining values and counters. Periods refer to the recurrence extended indefinitely, of which the prescribed schedule is a finite prefix.

2.2 Collision search with a bounded number of outputs

A bound on the number of outputs gives a collision search when the input set is sufficiently large. The following lemma accounts for both set sizes and allows nonuniform output probabilities.

Lemma 1 (Collision probability). *Let $\mathcal{D}$ be a finite nonempty set with $|\mathcal{D}| = D$, and let $f : \mathcal{D} \rightarrow \mathcal{Y}$ satisfy $|\text{Im}(f)| \leq K$ for a positive integer K . Draw q inputs independently and uniformly from $\mathcal{D}$, with $2 \leq q \leq K$. The probability of finding indices $i \neq j$ with distinct inputs $X_i \neq X_j$ and $f(X_i) = f(X_j)$ is at least*

$$1 - \exp\left(-\frac{q(q-1)}{2K}\right) - \frac{q(q-1)}{2D}. \quad (1)$$

Proof. Write the possible outputs as $y_1, \dots, y_m$, where $m \leq K$, and set

$$p_j = \Pr[f(X) = y_j] = \frac{|f^{-1}(y_j)|}{D}.$$

Pad this vector with zeros to length K . Its entries sum to one. Independence of the sampled inputs makes their outputs independent with this same probability vector.

For a fixed set of q different outputs, there are $q!$ possible orders, each with probability equal to the product of its q probabilities. Summing over all such sets gives

$$\Pr[\text{all outputs distinct}] = q! \sum_{1 \leq j_1 < \dots < j_q \leq K} p_{j_1} \cdots p_{j_q} = q! e_q(p_1, \dots, p_K).$$

To maximize this expression, fix two entries p_i, p_j and let v contain the remaining entries. Grouping terms according to whether they use neither, one, or both of these entries gives

$$e_q(p) = e_q(v) + (p_i + p_j)e_{q-1}(v) + p_i p_j e_{q-2}(v),$$

where $e_0 = 1$ and $e_k(v) = 0$ when k exceeds the length of v . All coefficients are nonnegative. For fixed sum $s = p_i + p_j$, the product satisfies $p_i p_j \leq s^2/4$, with equality at $p_i = p_j = s/2$. Thus averaging two entries cannot decrease e_q .

Choose a maximizer of e_q on the compact probability simplex that minimizes $\sum_j p_j^2$ among all maximizers. If two entries differed, averaging them would remain a maximizer and strictly decrease that sum, a contradiction. The maximum is therefore attained at $p_j = 1/K$ for every j . It follows that

$$\Pr[\text{all outputs distinct}] \leq q! \binom{K}{q} K^{-q} = \frac{(K)_q}{K^q} = \prod_{i=0}^{q-1} \left(1 - \frac{i}{K}\right)$$

where $(K)_q = K(K-1)\cdots(K-q+1)$. Applying $1-t \leq e^{-t}$ to each factor and using $\sum_{i=0}^{q-1} i = q(q-1)/2$ bounds this product by $\exp(-q(q-1)/(2K))$.

Let C be the event that some two outputs coincide, and let R be the event that some sampled input is repeated. Every outcome in $C \setminus R$ contains a collision between distinct inputs. For each pair of sample positions, $\Pr[X_i = X_j] = 1/D$; hence the union bound gives $\Pr[R] \leq \binom{q}{2}/D$. Therefore the desired probability is at least

$$\Pr[C \setminus R] \geq \Pr[C] - \Pr[R] \geq 1 - \exp\left(-\frac{q(q-1)}{2K}\right) - \frac{q(q-1)}{2D}.$$

The parameter K is any integer upper bound on the number of outputs. To derive the birthday estimate, suppose $K \geq 4$ is a square and take $q = \sqrt{K}$. Then $2 \leq q \leq K$, and the two terms become

$$\frac{q(q-1)}{2K} = \frac{1 - K^{-1/2}}{2}, \quad \frac{q(q-1)}{2D} = \frac{K - \sqrt{K}}{2D} < \frac{K}{2D}.$$

As K grows and K/D tends to zero, the lower bound tends to $1 - e^{-1/2} \approx 0.393469$. This is a constant success probability: it stays bounded below by a positive number

Algorithm 1 Collision search on a specified input set

Require: Map f that outputs a digest or a complete CV, finite input set $\mathcal{D}$, sample count q

```

1:  $L \leftarrow \emptyset$  ▷ One stored message for each observed output
2: for  $i = 1, \dots, q$  do
3:   Sample  $M$  uniformly from  $\mathcal{D}$ , independently of earlier samples
4:    $y \leftarrow f(M)$ 
5:   if  $L[y]$  is defined and  $L[y] \neq M$  then
6:     return  $(M, L[y])$ 
7:   end if
8:    $L[y] \leftarrow M$ 
9: end for
10: return failure

```

independent of the parameter size. For each parameter choice used below, the bound exceeds 0.39, as the following calculation shows.

In the CHIME, CHAMP, and QSH applications, $q \geq 2^{64}$, $q^2 = K$, and $q(q-1)/(2D) < 2^{-113}$. Consequently $q(q-1)/(2K) > 499/1000$ and $q(q-1)/(2D) < 1/1000$. For $0 < x < 1$, the alternating series gives

$$e^{-x} \leq 1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24}.$$

Substituting $x = 499/1000$ gives

$$P_{\text{coll}} > x - \frac{x^2}{2} + \frac{x^3}{6} - \frac{x^4}{24} - \frac{1}{1000} > \frac{391}{1000} > \frac{39}{100}. \quad (2)$$

Algorithm 1 makes at most q evaluations and stores at most q input–output pairs. It returns a pair when distinct inputs have the same output.

We measure cost in evaluations of the stated map and stored input–output pairs. For QSH this map outputs a complete leaf CV and reuses a fixed prefix; core inverse calls and finalization are counted separately. Maintaining a table indexed by output values also incurs comparison or lookup costs. If one run succeeds with probability at least $c > 0$, then t independent runs succeed with probability at least $1 - (1 - c)^t$. Reaching any fixed target success probability below one therefore requires at most a constant factor more evaluations.

For CHIME, the message family has $D = 2^{240}$ and at most $K = 2^{128}$ outputs. For CHAMP, messages of length $4w$ give $D = 2^{4w}$, while $K = 2^{3w}$ bounds the number of matrices with the required determinant. These choices make K/D small in both applications. For QSH’s periodic families, $D = 2^{8rw}$, $K = 2^{4rw}$, and $q = 2^{2rw}$, with $w \in \{32, 64\}$, $r \in \{1, 2\}$. The family with an unrestricted final block in a complete data chunk uses $D = 2^{32w}$, $K = 2^{16w}$, and $q = 2^{8w}$. In each case $D = K^2$, so the term accounting for repeated sampled inputs is less than $1/(2K) \leq 2^{-129}$. Thus the same rational bound (2) applies. All these searches use independent uniform samples with replacement and require the returned inputs to differ.

3 MoFang: Collisions from Equivalent Message Blocks

Differences in MoFang’s two message branches cancel at the initial key injection and after each of the sixteen expansion steps. A four-dimensional space of such differences gives at least sixteen equivalent blocks for every incoming chaining state. We use this equivalence to construct collisions for all four variants with fixed output lengths and both XOF variants.

3.1 Word positions preserved by message expansion

A message block consists of two halves, each represented as a 3×3 array of 64-bit words. Index the words by $i = 3x + y$, where $x, y \in \mathbb{Z}_3$. Both permutations in the message expansion exchange positions

$$(1, 2) \longleftrightarrow (2, 1), \quad \text{equivalently } 5 \longleftrightarrow 7.$$

For an array A , the two permutations are defined by

$$(P_0 A)[x, y] = A[y, 2x + y], \quad (P_1 A)[x, y] = A[x + 2y, x],$$

with coordinates reduced modulo three. In the order $i = 0, \dots, 8$, the input indices assigned to the output positions are respectively

$$(0, 4, 8, 2, 3, 7, 1, 5, 6), \quad (0, 6, 3, 4, 1, 7, 8, 5, 2).$$

The nonlinear operation changes only row zero, before the permutation. Neither position 5 nor position 7 enters that row; the round constant is injected at position zero [2, Section 2.4.4, Tables 5–6, pp. 12–15]. The xor of the message branches forms the nine-word value injected into the data state.

We use the branch equations with left rotations below [2,12]. For the two message branches, the rotation amounts at output positions (5, 7) are (31, 47) and (61, 29), respectively. The 768- and 1024-bit modes also update a branch derived from the incoming chaining value; that branch is identical in the two evaluations. Both compression modes execute sixteen data rounds after the initial key injection [2, Table 1 and Algorithms 5–6]. If B_r, C_r denote the expanded message halves and Z_r the optional chaining branch, then

$$K_r = B_r \oplus C_r \quad \text{or} \quad K_r = Z_r \oplus B_r \oplus C_r, \quad 0 \leq r \leq 16.$$

The chaining branch evolves independently of the message. The data path has the form

$$X_0 = P \oplus K_0, \quad X_{r+1} = F_r(X_r) \oplus K_{r+1}, \quad 0 \leq r < 16.$$

Here F_r is the round function acting on the data state. The argument applies to arbitrary deterministic F_r and to any nonlinear map confined to row zero of each message branch.

Let C_{DM} and C_{DBL} denote the single- and double-block-length compression functions. Their chaining states are, respectively, an array S of nine words and an ordered pair (L, R) of such arrays; both take a message block of eighteen words. Let e_c be the vector of nine words with word 8 equal to c and all other words zero. Writing $E_M(P)$ for the sixteen-round cipher gives

$$\begin{aligned} C_{\text{DM}}(S, M) &= E_M(S) \oplus S, \\ C_{\text{DBL}}((L, R), M) &= (E_{R,M}(L) \oplus L, \\ &\quad E_{R,M}(L \oplus e_1) \oplus (L \oplus e_1)). \end{aligned} \quad (3)$$

The subscript R, M includes the chaining branch R in the second mode. These equations include the feed-forward in both output components.

Define the two-dimensional word space

$$\begin{aligned} \mathcal{P} &= \{u \in \{0, 1\}^{64} : \text{rotl}_{64}(u, 2) = u\} \\ &= \{0000000000000000, 5555555555555555, \\ &\quad \text{aaaaaaaaaaaaaaaa, ffffffffffffffffff}\}. \end{aligned} \quad (4)$$

To see its dimension, number a word's bits modulo 64. The condition $\text{rotl}_{64}(u, 2) = u$ identifies all even-indexed bits with each other and all odd-indexed bits with each other. The two parity classes are independent, giving $2^2 = 4$ words: the four displayed masks. For $a, b \in \mathcal{P}$, let $\Delta(a, b)$ be the eighteen-word difference supported at positions 5, 7, 14, 16, with respective values a, b, a, b . Thus the two message halves receive the same difference.

Proposition 1 (Equivalent message blocks). *For every message block M , every $a, b \in \mathcal{P}$, and all nine-word states S, L, R ,*

$$\begin{aligned} C_{\text{DM}}(S, M) &= C_{\text{DM}}(S, M \oplus \Delta(a, b)), \\ C_{\text{DBL}}((L, R), M) &= C_{\text{DBL}}((L, R), M \oplus \Delta(a, b)). \end{aligned}$$

Proof. Write d_r and $\tilde{d}_r$ for the xor differences between the two evaluations in the first and second message branches after r expansions. Initially both equal the nine-word vector whose positions (5, 7) contain (a, b) and whose other positions are zero. The initial injection difference is therefore $d_0 \oplus \tilde{d}_0 = 0$. The branch derived from the incoming chaining value, when present, also has zero difference.

Suppose that $d_r = \tilde{d}_r$ at step r , with support contained in $\{5, 7\}$. Write

$$a_r = (d_r)_5, \quad b_r = (d_r)_7, \quad (a_0, b_0) = (a, b),$$

and assume $a_r, b_r \in \mathcal{P}$. Since these positions bypass the S-box, their next differences in the two branches are

$$\begin{aligned} &(\text{rotl}_{64}(b_r, 31), \text{rotl}_{64}(a_r, 47)), \\ &(\text{rotl}_{64}(b_r, 61), \text{rotl}_{64}(a_r, 29)). \end{aligned}$$

No other word acquires a difference: the S-box reads equal first rows, and both word permutations exchange positions 5 and 7. For $u \in \mathcal{P}$ and an odd amount $2t + 1$,

$$\text{rotl}_{64}(u, 2t + 1) = \text{rotl}_{64}(\text{rotl}_{64}(u, 2t), 1) = \text{rotl}_{64}(u, 1).$$

Rotating a word once exchanges its two parity classes, so it remains in $\mathcal{P}$. Both next branch differences therefore have the pair

$$(a_{r+1}, b_{r+1}) = (\text{rotl}_{64}(b_r, 1), \text{rotl}_{64}(a_r, 1)) \in \mathcal{P}^2. \quad (5)$$

Two applications rotate each mask by two bits and hence give $(a_{r+2}, b_{r+2}) = (a_r, b_r)$. The two branch differences remain equal and cancel in the injection. The round constant has zero difference because it is the same in both evaluations. This closes the induction. Starting at $r = 0$ and iterating through $r = 16$ gives equality of all seventeen injections.

The data states start equal and receive identical injections in every round, so the cipher outputs agree. Feed-forward uses the same incoming chaining value. For C_{DBL} , the argument applies separately to both cipher calls, whose plaintexts differ by a fixed domain bit between calls but agree between the two message evaluations. Both output halves therefore agree. $\square$

The mask restriction already follows from the first expansion. Equality of the two branch differences at position 5 requires $\text{rotl}_{64}(b, 31) = \text{rotl}_{64}(b, 61)$, or $b = \text{rotl}_{64}(b, 30)$. Equality at position 7 similarly requires $a = \text{rotl}_{64}(a, 18)$. On 64 bit positions, adding 30 or adding 18 has exactly $\gcd(64, 30) = \gcd(64, 18) = 2$ orbits, namely the two parity classes. Thus the masks a, b in $\Delta(a, b)$ must lie in $\mathcal{P}$ for the branch differences to agree. The specification uses right rotations and lists the two-dimensional rotation tables transposed relative to the C arrays. With either convention, all four rotation amounts are odd and act identically on $\mathcal{P}$.

3.2 Collision pairs and equivalent messages

The four variants with fixed output lengths pad with one bit followed by the fewest zero bits needed to reach a multiple of 1152 bits. With the default initialization, the 256- and 512-bit variants start from e_5 and e_6 , respectively; the 768- and 1024-bit variants start from $(e_{10}, 0^{576})$ and $(e_{11}, 0^{576})$ [2, Table 7 and Algorithms 5–6]. Before the final compression, the 256- and 512-bit variants xor e_1 into S ; the 768- and 1024-bit variants xor e_2 into L .

We parse and serialize words in little-endian order, as in the submitted implementations [12]. Let $\text{out}_j(A)$ be the serialization of the first j words of A , reading bits from most to least significant within each byte. The four output maps are

$$\begin{aligned} O_{256}(S) &= \text{out}_4(S), & O_{512}(S) &= \text{out}_8(S), \\ O_{768}(L, R) &= \text{out}_6(L) \parallel \text{out}_6(R), \\ O_{1024}(L, R) &= \text{out}_8(L) \parallel \text{out}_8(R). \end{aligned} \quad (6)$$

Corollary 1 (Equivalent messages for fixed output lengths). *For each of these four variants and any message containing $q \geq 1$ complete 144-byte original blocks, independent choices of $(a, b) \in \mathcal{P}^2$ in each block produce at least*

$$16^q = 2^{4q}$$

distinct equal-length messages with one common digest.

Proof. Each block has $|\mathcal{P}|^2 = 4^2 = 16$ mask choices. Different choices give distinct blocks because $M \oplus \Delta = M \oplus \Delta'$ implies $\Delta = \Delta'$, which recovers both masks at positions 5 and 7. Choices in different original blocks affect disjoint byte positions, so their counts multiply. The changes preserve the message length and padding. Starting from the common initial state, equality of the chaining values follows by induction over blocks: equal incoming values and equivalent current blocks give equal outgoing values by Proposition 1. The unchanged tail and padding give identical remaining blocks. The prescribed xor of e_1 or e_2 before the final compression also preserves equality of the incoming states. Applying the variant's output map then gives equal digests. A second message is obtained by applying a nonzero mask pair to any one of the complete original blocks.

A shorter pair changes only position 5 in each half and leaves the padding untouched. With exponents denoting byte repetition, set

$$M = 00^{120}, \quad M' = 00^{40} \parallel \mathbf{ff}^8 \parallel 00^{64} \parallel \mathbf{ff}^8. \quad (7)$$

Both are 120-byte unpadded messages whose words are either zero or $2^{64} - 1$, so their byte representations are independent of endianness. Each receives the 24-byte suffix $80 \parallel 00^{23}$. The changed byte ranges are 40–47 and 112–119, exactly words 5 and $14 = 9 + 5$. They instantiate $a = 2^{64} - 1$, $b = 0$. Both ranges precede the padding delimiter at byte 120, so the completed 144-byte blocks differ by precisely the proved mask. For each variant, applying its output function to the equal chaining states gives equal digests for M and M' .

A 136-byte base message accommodates all four modified word positions before its padding delimiter. Its eight-byte padding is $80 \parallel 00^7$, which remains unchanged under all sixteen choices of (a, b) . The resulting messages are distinct and their padded blocks differ by allowed masks, so Proposition 1 gives the same digest for all sixteen messages in each of the four variants.

3.3 Collisions for the XOF variants

Proposition 1 applies to every incoming chaining state. We now use this property under the XOF initializations and at each subsequent squeezing step. Write $B \sim B'$ when $B' = B \oplus \Delta(a, b)$ for some $a, b \in \mathcal{P}$; this includes $B = B'$ by taking $a = b = 0$. Since $\mathcal{P}$ is closed under xor and Δ is linear, any two variants of the same block in Section 3.2 are equivalent.

The two XOFs use the padding of Section 3.2. Their default initial states are e_7 for 256-XOF and $(e_{15}, 0^{576})$ for 768-XOF. Each absorbs nonfinal blocks with

its corresponding compression function. Before the final compression, 256-XOF xors e_1 into its state and 768-XOF xors e_2 into the left component. Let S_0 and $V_0 = (L_0, R_0)$ denote the respective states after absorption, and let B_* denote the last padded message block before expansion.

Writing $V_t = (L_t, R_t)$, the squeezing transitions have the form

$$\begin{aligned} S_{t+1} &= C_{\text{DM}}(S_t \oplus e_1, Q), \\ V_{t+1} &= C_{\text{DBL}}(V_t \oplus \eta, 0^{1152}), \quad t \geq 0, \end{aligned} \tag{8}$$

where xor on a state pair is componentwise. The design document [2, Algorithms 5–6 and Section 2.8] and the submitted reference and optimized implementations [12] use the following parameters:

Parameter	Design document	Both implementations
256-XOF: Q	0^{1152}	B_*
768-XOF: η	$(e_2, 0^{576})$	$(0^{576}, e_2)$

The block B_* is unchanged during squeezing: compression expands a copy of its message input. The proof below covers both choices of Q and both choices of η .

For fixed Q and η , define output blocks by

$$Y_t^{(256)} = O_{256}(S_t), \quad Y_t^{(768)} = O_{768}(L_t, R_t).$$

Thus the first output block is extracted immediately after absorption. For $v \in \{256, 768\}$ and a finite positive output length ℓ_{out} , let $\text{XOF}_{v, \ell_{\text{out}}}(M)$ be the first ℓ_{out} bits of $Y_0^{(v)} \parallel Y_1^{(v)} \parallel \dots$. This also specifies the final partial output block.

Corollary 2 (XOF collisions). *Let M, M' be the pair in (7), or any two distinct members of one of the message families in Section 3.2. Then*

$$\text{XOF}_{v, \ell_{\text{out}}}(M) = \text{XOF}_{v, \ell_{\text{out}}}(M')$$

for each $v \in \{256, 768\}$, each squeezing convention above, and every finite positive ℓ_{out} . The same message pair applies to all these output lengths.

Proof. Fix one XOF variant and a message pair from Section 3.2. Use primes for the evaluation on M' . The messages have the same length and corresponding padded blocks are equivalent under $\sim$. They start from the same XOF initial state. Equal incoming states and equivalent blocks give equal outgoing states by Proposition 1. Before the final compression, both evaluations xor e_1 into their states for 256-XOF, or e_2 into their left components for 768-XOF, preserving equality. Induction over the padded blocks therefore gives $S_0 = S'_0$ for 256-XOF and $V_0 = V'_0$ for 768-XOF.

If only complete original blocks were modified, the final padded blocks are equal, including when padding adds a new block. For the 120-byte pair and the 136-byte families, the final padded blocks may differ, but their difference is $\Delta(a, b)$. Thus in every case $B_* \sim B'_*$.

For 256-XOF with $Q = B_*$, suppose $S_t = S'_t$. Then

$$\begin{aligned} S_{t+1} &= C_{\text{DM}}(S_t \oplus e_1, B_*) \\ &= C_{\text{DM}}(S_t \oplus e_1, B'_*) \\ &= C_{\text{DM}}(S'_t \oplus e_1, B'_*) = S'_{t+1}. \end{aligned}$$

The second equality follows from Proposition 1 at the state $S_t \oplus e_1$, since $B_* \sim B'_*$; the third uses the induction hypothesis. When $Q = 0^{1152}$, both calls have the same message input, so equal states also give equal successors. In either case, induction yields $S_t = S'_t$ for every $t \geq 0$.

For 768-XOF, fix either value of η in the table. Whenever $V_t = V'_t$, we have $V_t \oplus \eta = V'_t \oplus \eta$, and both evaluations use the message block 0^{1152} in (8). Deterministic compression gives $V_{t+1} = V'_{t+1}$, and induction yields equality of all subsequent state pairs.

The output maps consequently give $Y_t^{(v)} = Y'^{(v)}_t$ for every t . Concatenating these equal blocks and taking the requested prefix proves the claim for every finite positive output length. $\square$

4 Neulaser: Collisions from a Noninjective State Update

We construct collisions for Neulaser-512, Neulaser-768, and Neulaser-1024 by choosing messages whose internal states become equal after the first update. We analyze the reference and optimized implementations accompanying the Neulaser v2 submission dated 30 June 2026 [13]. Both load $V \parallel K$ directly as consecutive big-endian 32-bit words, with no initial reduction modulo p . This supplies the mapping from PRNG inputs to state words left unspecified by the design interface [14, Secs. 3.1–3.3, pp. 6–9]. The collisions use the prescribed IVs, padding, and all initialization and output rounds.

4.1 Two inputs with equal feedback

Let $\mathcal{W}_{32} = \{0, \dots, 2^{32} - 1\}$ and $\mathcal{C}_p = \{0, \dots, p - 1\}$, where $p = 4294967291$. Each module contains sixteen words $L = (s_0, \dots, s_{15}) \in \mathcal{W}_{32}^{16}$. Bitwise operations act on their 32-bit encodings; arithmetic modulo p returns a representative in $\mathcal{C}_p$. The local feedback is

$$\ell_{\text{lin}}(L) = 8193s_0 + 16777217s_4 + 4194304s_9 + 9s_{13} + 524288s_{15} \pmod{p}, \quad (9)$$

$$\Phi(L) = \text{red}_p(\ell_{\text{lin}}(L) + F(s_{15}, s_{10}, s_5, s_0)). \quad (10)$$

Here red_p is reduction to the least nonnegative residue modulo p . The nonlinear function F is defined as follows. Set

$$A = R_0 + R_1 \pmod{2^{32}}, \quad B = R_2 \oplus R_3, \quad U_F \parallel V_F = \text{rotl}_{64}(A \parallel B, 16).$$

Here U_F, V_F are the two 32-bit halves of the rotated word. For a byte $x = \sum_{i=0}^7 x_i 2^i$, identify x with $\sum_{i=0}^7 x_i z^i$ in $\mathbb{F}_2[z]/(z^8 + z^5 + z^3 + z^2 + 1)$. Let $v = 0$ when $x = 0$, and $v = x^{-1}$ in this field otherwise. The byte substitution is

$$S(x) = v \oplus \text{rotr}_8(v, 1) \oplus \text{rotr}_8(v, 3) \oplus \text{rotr}_8(v, 4) \oplus \text{rotr}_8(v, 6) \oplus \mathbf{a7}. \quad (11)$$

The operator $\mathcal{S}$ applies S independently to the four bytes of a 32-bit word. Define

$$L_1(X) = \bigoplus_{j \in \{0, 2, 10, 18, 24\}} \text{rotl}_{32}(X, j), \quad L_2(X) = \bigoplus_{j \in \{0, 8, 14, 22, 30\}} \text{rotl}_{32}(X, j).$$

Then

$$F(R_0, R_1, R_2, R_3) = ((\mathcal{S}(L_1(U_F)) \oplus \text{rotl}_{32}(R_1, 8)) + \mathcal{S}(L_2(V_F))) \bmod 2^{32}.$$

Thus the arithmetic in the nonlinear layer is modulo 2^{32} , whereas the final local reduction is modulo p .

Restricting to $L_x = (x, 0, \dots, 0)$ gives

$$\Phi(L_x) = (8193x + F(0, 0, 0, x)) \bmod p.$$

The two values $x = 00002743$ and $x = 0002\text{ad}84$ give the following intermediate words. Each column can be evaluated using (11) and the rotations above.

Quantity	$x = 00002743$	$x = 0002\text{ad}84$
U_F	00000000	00000002
V_F	27430000	ad840000
$L_1(U_F)$	00000000	0208080a
$L_2(V_F)$	ad9a1937	02ce4acc
$\mathcal{S}(L_1(U_F))$	a7a7a7a7	9b4e4e01
$\mathcal{S}(L_2(V_F))$	dfb1ac7e	9b405fe3

Because $R_1 = 0$, adding the last two entries of each column modulo 2^{32} gives the values of $F(0, 0, 0, x)$ below. The two inputs have equal feedback:

x	$8193x \bmod p$	$F(0, 0, 0, x)$
00002743	04e88743	87595425
0002ad84	55b32d84	368eade4

Adding the two columns in each row gives

$$\begin{aligned} 04\text{e}88743 + 87595425 &= 8\text{c}41\text{db}68, \\ 55\text{b}32\text{d}84 + 368\text{eade}4 &= 8\text{c}41\text{db}68. \end{aligned}$$

Both sums equal $8\text{c}41\text{db}68$, which is below $p = \text{ffffffffffb}$ and is therefore also their residue modulo p . The two distinct canonical inputs have the same local feedback: the change in the linear term is canceled by the change in F before reduction.

Finding the two constants. Define $f(x) = \Phi(L_x)$. Enumerate $x = 0, 1, \dots$, storing $(f(x), x)$ indexed by the feedback value. The first repeated value occurs at $x = 0002\text{ad}84$, with the earlier input 00002743 . Thus deterministic enumeration finds the pair after $N = 175493$ local feedback evaluations. A balanced search tree uses $O(N \log N)$ word comparisons and $O(N)$ records.

4.2 Equality of the states after one update

Write $s_{i,j}$ for word j of module i , with module indices modulo the number k of modules. Appendix A gives the complete equations for the confusion vector Γ_r ; they read only positions 14, 9, 6, and 1. Define

$$(\delta_j, t_j) = \begin{cases} (1, 5), & j = 3, \\ (2, 13), & j = 7, \\ (1, 21), & j = 11, \ k = 3, \\ (3, 21), & j = 11, \ k \in \{4, 5\}. \end{cases}$$

The simultaneous state update is

$$s'_{i,j} = \begin{cases} s_{i,j+1}, & 0 \leq j < 15, \ j \notin \{3, 7, 11\}, \\ \text{red}_p(s_{i,j+1} \oplus \text{rotl}_{32}(\Gamma_r^{(i+\delta_j)}, t_j)), & j \in \{3, 7, 11\}, \\ \text{red}_p(\Phi(L_i) \oplus \Gamma_r^{(i)}), & j = 15. \end{cases} \quad (12)$$

This update maps $\mathcal{W}_{32}^{16k}$ to itself and preserves $\mathcal{C}_p^{16k}$: each output either copies a word or reduces an integer modulo p . The shifts copy positions 1 through 15, so the old $s_{i,0}$ contributes to the next state only through $\Phi(L_i)$.

Proposition 2 (Equal states after one update). *Let two states in $\mathcal{W}_{32}^{16k}$ agree everywhere except at position 0 of one module, and suppose that the local feedbacks of that module agree. At the same round number, their next complete states are equal, for each supported module count. In particular, the module contents $L_{00002743}$ and $L_{0002\text{ad}84}$ satisfy this condition.*

Proof. The differing coordinate is absent from every input to Γ_r , so the global confusion vectors agree. In the affected module, the two local feedback values agree by hypothesis. All other modules have identical local feedbacks. The ordinary shifts read positions 1 through 15, which are equal; the internal injections read equal confusion values; and the new words at position 15 are equal. Every coordinate of the next state agrees.

Once the states are equal, subsequent blank rounds preserve equality. We next construct messages that initialize the two states under direct loading.

4.3 Collisions for the three hash variants

Initialization concatenates the chaining vector V and message seed K , parses the result into big-endian 32-bit words, and fills the modules in order. Let $\text{BE}_t(x)$ denote the big-endian encoding of x in t bits.

For digest length n , the chaining vector has $n + 64$ bits. If there are k modules, their total size is $16 \cdot 32k = 512k$ bits. Direct loading therefore assigns $512k - (n + 64)$ bits to the seed. In bytes, its length and the offset of the last module's first word are respectively

$$B_n/8 = 64k - \frac{n + 64}{8}, \quad d_n = 64(k - 1) - \frac{n + 64}{8}.$$

The last module starts at byte $64(k - 1)$ of the concatenation. Subtracting the $(n + 64)/8$ bytes occupied by V gives its offset within K . Substituting the three parameter pairs $(n, k) = (512, 3), (768, 4), (1024, 5)$ gives the table below. Here B_n is the message block length in bits, and d_n is the byte offset within the message block:

n	k	$B_n/8$	d_n
512	3	120	56
768	4	152	88
1024	5	184	120

To specify the hash mode, write $v = n + 64$ and let $K_0, \dots, K_{\ell-1}$ be the padded blocks. Define $C_j = 0^{v-64} \parallel \text{BE}_{64}(j)$. Each PRNG invocation starts its round index at zero, performs 32 blank updates, and then produces output as in Appendix A. Let Z_j be the output of the invocation on (V_j, K_j) , of length v for a nonfinal block and $v + n$ for the final block. Write $Z[a : b]$ for bits $a, \dots, b - 1$ of Z , numbered from zero. The mode is

$$\begin{aligned} V_0 &= IV_n \oplus C_0, \\ V_{j+1} &= Z_j[0 : v] \oplus C_{j+1}, \quad 0 \leq j < \ell - 1, \\ H_n(M) &= Z_{\ell-1}[v : v + n]. \end{aligned} \tag{13}$$

The prescribed IVs are given in Appendix A.

Corollary 3 (Collisions for Neulaser). *For $n \in \{512, 768, 1024\}$, set $a = 00002743$, $b = 0002ad84$, and*

$$M_{n,x} = 00^{d_n} \parallel \text{BE}_{32}(x) \parallel 00^{60}, \quad x \in \{a, b\},$$

where the exponents count bytes. Under the loading rule and hash mode above, with the prescribed IV, padding, and all rounds,

$$M_{n,a} \neq M_{n,b}, \quad H_n(M_{n,a}) = H_n(M_{n,b}).$$

Proof. The word at byte offset d_n becomes position 0 of the last module. The following 60 zero bytes fill its other fifteen words. The two initial states therefore have last modules L_a and L_b , and agree everywhere else. Proposition 2 gives

equal complete states after the first blank update. The remaining blank updates and the output updates use the same round numbers and preserve equality. The outputs from processing the first block therefore agree, as do the two values of V_1 .

Each original message has exactly B_n bits. Its common padding block is

$$P_n = 80 \parallel 00^{B_n/8-9} \parallel \text{BE}_{64}(B_n), \quad B_n \in \{960, 1216, 1472\},$$

again with byte repetition. The final PRNG calls have identical chaining vectors and seeds P_n , so their streams agree. Equation (13) then gives equal n -bit digests. The original messages are distinct because $a \neq b$.

With the constants fixed, each message is constructed by placing its nonzero word at offset d_n in an otherwise zero block. Evaluating a pair costs two hash calls. Appendix A gives the three common digests. Since the proof uses only equality of the initial chaining vectors, the construction applies to any common value of V_0 .

5 CHIME-512: Collision Search Using an Invariant Subspace

We construct 2^{240} messages whose CHIME-512 digests take at most 2^{128} values when shifts act independently on 64-bit words, as in the submitted reference and optimized implementations. Sampling from this family finds a collision with probability greater than 0.39 using at most 2^{64} hash evaluations. Each evaluation absorbs two blocks and executes all 15 rounds after each block. For comparison, a birthday search on an ideal 512-bit hash requires about 2^{256} evaluations for the same success probability.

5.1 State and round operations

CHIME uses six 256-bit arrays $(A_0, A_1, B_0, B_1, C_0, C_1)$. The 512-bit instance starts from zero, absorbs blocks into A_0, A_1 , applies 15 rounds per block, and returns the first 512 state bits. Its padding appends $1 \parallel 0^k \parallel 1$, with the fewest zero bits needed to reach a multiple of 512 bits [4, pp. 3–7]. Appendix B specifies every round operation, the constant formula, and the word permutations.

The notation section explicitly defines left shifts and rotations independently on each 64-bit word. The linear layer also contains $C_1 \leftarrow C_1 \oplus (A_1 \gg 5)$, but the symbol $\gg$ has no separate width definition. Throughout this section, that operation applies a logical right shift separately to each 64-bit word. Both submitted implementations use this operation [5]. They parse and serialize words in little-endian order, whereas the specification uses big-endian order. The proof below applies to both byte orders; Section 5.5 distinguishes their numerical outputs.

5.2 An invariant subspace

Identify a 64-bit word x with its 64-bit binary representation, written from most to least significant bit, and define

$$W(x) = x \parallel x \parallel x \parallel x$$

and the 384-dimensional binary subspace

$$\mathcal{U} = \{(W(x_0), \dots, W(x_5)) : x_i \in \{0, 1\}^{64}\}.$$

Proposition 3 (Round invariance). *With shifts applied separately to each 64-bit word, every specified round maps $\mathcal{U}$ into $\mathcal{U}$.*

Proof. The map W copies its argument into four lanes. For any words x, y , for example,

$$W(x) \oplus W(y) = W(x \oplus y), \quad W(x) \wedge W(y) = W(x \wedge y), \quad \neg W(x) = W(\neg x).$$

A rotation or logical shift applied separately to each 64-bit word gives four copies of the rotated or shifted word. Thus XOR, AND, complement, and the specified rotations and shifts all preserve this form. The operations Π_{perm64} , Π_{swap64} , and Π_{swap128} exchange identical words or identical word pairs, so they fix $W(x)$.

Write $x = a \parallel b$ with 32-bit halves. The specified Π_{perm32} exchanges their roles within $W(x)$:

$$\Pi_{\text{perm32}}(W(x)) = W(\text{rotl}_{64}(x, 32)).$$

Thus it also produces an array of the form $W(y)$. The xor updates in the linear layer preserve this form by the first identity above.

Each round constant is four copies of one 64-bit word, so XORing it into A_0 preserves $\mathcal{U}$. In each round, the nonlinear layer applies the same Boolean functions at corresponding bit positions in all four words. Permuting the six arrays also preserves their form. Every layer therefore preserves $\mathcal{U}$.

Increasing the number of rounds preserves the invariant whenever the additional rounds use the same operations and constants of this form.

5.3 Messages that preserve the subspace

Choose arbitrary $a, b, c \in \{0, 1\}^{64}$ and $d \in \{0, 1\}^{48}$. Form the 1024-bit string

$$Q = W(a) \parallel W(b) \parallel W(c) \parallel W(d \parallel 8001), \quad (14)$$

where 8001 denotes a 16-bit string. Let $M(a, b, c, d)$ be Q with its final 16 bits removed. The parameters are bit strings in this construction; grouping successive bits into bytes, with the most significant bit first, fixes the external message bytes. These same bytes are supplied to either implementation, independently of the byte order used to load its 64-bit words.

Proposition 4 (Messages and their digests). *The messages $M(a, b, c, d)$ form a family of 2^{240} distinct 126-byte messages. Their standard padding is exactly Q , and their CHIME-512 digests have the form $W(u) \parallel W(v)$.*

Proof. The message length is $1024 - 16 = 1008$ bits. Padding to a multiple of the 512-bit rate requires 16 bits, and $10^{14}1$ is precisely 8001. Hence padding recovers Q . The parameters a, b, c occur in complete repeated blocks, and d occurs in complete words before the removed suffix; distinct parameter tuples therefore give distinct messages.

The initial state is in $\mathcal{U}$. The first rate block injects $W(a), W(b)$ into A_0, A_1 , preserving membership. Proposition 3 preserves membership during all 15 rounds. The second block injects $W(c), W(d \parallel 8001)$, and the same argument applies. The final arrays A_0, A_1 each have the form $W(x)$, giving the stated 512-bit digest form.

The message family has $64 + 64 + 64 + 48 = 240$ independent bits. Each digest $W(u) \parallel W(v)$ is determined by the two 64-bit words u, v , giving at most $2^{64+64} = 2^{128}$ outputs. The output map selects two of the six arrays, so the digest is determined by 128 of the 384 bits that describe a state in $\mathcal{U}$.

The same byte strings give the same output bound under the submitted implementations' little-endian convention. Let β reverse the eight bytes of a 64-bit word. Each repeated array $W(x)$ then loads as $W(\beta(x))$, so both absorbed blocks still belong to the required set. On this set, the implementations' 32-bit word permutation swaps the two halves of the repeated word, and their other word permutations fix each array, just as in Proposition 3. The remaining round operations are the stated wordwise operations. The state therefore stays in $\mathcal{U}$, and serializing the two output words in little-endian order still gives at most 2^{128} digests. The message count and the padding argument are unchanged.

5.4 Collision search and success probability

Sample a, b, c, d uniformly and independently. The injective message construction makes this uniform sampling from the legal family. In Lemma 1, set $D = 2^{240}$, $K = 2^{128}$, and $q = 2^{64}$, which satisfies $2 \leq q \leq K$. Substitution gives

$$\frac{q(q-1)}{2K} = \frac{1-2^{-64}}{2}, \quad \frac{q(q-1)}{2D} < \frac{2^{128}}{2 \cdot 2^{240}} = 2^{-113}.$$

Thus the probability of a collision between distinct original messages is

$$P_{\text{coll}} \geq 1 - \exp\left(-\frac{1-2^{-64}}{2}\right) - 2^{-113} > 0.39. \quad (15)$$

The lower bound is close to $1 - e^{-1/2}$. Algorithm 1 samples (a, b, c, d) , constructs $M(a, b, c, d)$, and compares the resulting digests. It succeeds with probability greater than 0.39 using at most 2^{64} hash evaluations and 2^{64} stored records. Each evaluation makes two calls to the 15-round permutation, with either byte order defined above.

5.5 Example

For $a = 1, b = 2, c = 3, d = 0$, the padded message consists of $W(1) \parallel W(2)$ and $W(3) \parallel W(0000000000008001)$. Absorbing these blocks from the zero state and applying rounds $r = 0, \dots, 14$ after each block, as defined in Appendix B, with big-endian word loading and serialization gives the digest

$$W(2a5691178e43486c) \parallel W(19a41795fd802a99).$$

For the same message bytes, little-endian loading replaces each absorbed word x by $\beta(x)$. Applying the same round equations on $\mathcal{U}$ and serializing the final words in little-endian order, as in both submitted implementations, gives

$$\text{ef3be6a284ddbfa8}^4 \parallel \text{36f4af87fc2dc4fa}^4.$$

Here each hexadecimal string denotes eight bytes in reading order, and the exponent denotes four repetitions of that string.

6 CHAMP: Collision Search Using a Determinant Constraint

For messages of a fixed length, CHAMP's matrix products all have the same determinant. Counting the matrices with this determinant bounds the number of possible digests. We use this bound to obtain a collision search using at most $2^{3w/2}$ hash evaluations with success probability greater than 0.39, compared with the birthday cost $O(2^{2w})$ for an ideal $4w$ -bit hash.

6.1 The determinant of a message product

For $w \in \{128, 256\}$, the specification uses

$$p = \begin{cases} 2^{128} - 15449, & w = 128, \\ 2^{256} - 36113, & w = 256. \end{cases}$$

A message bit selects either A or B , and the intermediate hash is their ordered product over $\mathbb{F}_p$, starting from the identity [3, pp. 3–5]. The proposed choices are

$$A = \begin{pmatrix} -2 & 1 \\ 4 & -3 \end{pmatrix}, \quad B_1 = \begin{pmatrix} -5 & 2 \\ -6 & 2 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 6 & 4 \\ -2 & -1 \end{pmatrix}.$$

All three generators have determinant two:

$$\det A = \det B_1 = \det B_2 = 2.$$

For a message $M = m_1 \cdots m_L$, write $G_0 = A$ and let G_1 be the selected B . Starting from the identity gives $H_{\text{mat}}(M) = G_{m_1} \cdots G_{m_L}$. Multiplicativity of the determinant now yields, for either pair,

$$\det H_{\text{mat}}(M) = \prod_{i=1}^L \det G_{m_i} = 2^L \bmod p. \quad (16)$$

This is a constraint on the matrix product, before the final entrywise inversion.

The finalizer replaces each entry x by

$$\text{inv}_0(x) = \begin{cases} x^{-1}, & x \neq 0, \\ 0, & x = 0, \end{cases}$$

and serializes four words of w bits each in the order $(1, 1), (2, 1), (1, 2), (2, 2)$. For $x \neq 0$, applying inv_0 twice gives $(x^{-1})^{-1} = x$; zero is fixed. Hence inv_0 is an involution, and in particular a bijection, on $\mathbb{F}_p$. Each field element has a unique canonical integer representative in $0, \dots, p-1$, which fits into w bits because $p < 2^w$. The fixed order and fixed word width therefore make serialization injective. Applying these maps to all four entries preserves the cardinality of every intermediate image.

Proposition 5 (Number of digests for a fixed message length). *For either proposed generator pair and every fixed length L , CHAMP produces at most $p(p^2 - 1)$ distinct digests on messages of length L .*

Proof. Since p is odd, $\delta = 2^L \bmod p$ is nonzero. In a matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with determinant δ , the first row (a, b) must be nonzero, giving $p^2 - 1$ choices. For any such row, $ad - bc = \delta$ is a nonzero linear equation in (c, d) , with exactly p solutions over $\mathbb{F}_p$. Indeed, if $a \neq 0$, one may choose any $c \in \mathbb{F}_p$ and must then set $d = a^{-1}(\delta + bc)$. If $a = 0$, then $b \neq 0$, so $c = -b^{-1}\delta$ is forced while d is arbitrary. Each case has exactly p choices for the second row. There are therefore $p(p^2 - 1)$ matrices with determinant δ . Every message product belongs to this set by (16), and injective finalization preserves the number of distinct values.

6.2 Collision search and success probability

The hash processes exactly L message bits and appends no padding [3, Listing 1]. To evaluate it, multiply the selected generators in message order, apply inv_0 to each entry, and serialize the four canonical integers as w -bit big-endian words. Choose messages of length $L = 4w$, and sample their bits independently and uniformly. The input set is therefore $\mathcal{D} = \{0, 1\}^{4w}$, with $D = 2^{4w}$. All sampled messages have the same determinant $2^{4w} \bmod p$, so the preceding proposition bounds their number of possible digests by

$$K_{\text{det}} = p(p^2 - 1) = p^3 - p < p^3 < 2^{3w}.$$

The strict inequalities use $p > 0$ and $p < 2^w$. Apply Lemma 1 using the slightly looser integer bound $K_0 = 2^{3w}$ as its parameter K , and take

$$q = 2^{3w/2}, \quad q^2 = K_0.$$

For both specified word widths, q is an integer satisfying $2 \leq q \leq K_0$, and

$$\frac{q(q-1)}{2K_0} = \frac{q^2(1-q^{-1})}{2q^2} = \frac{1-2^{-3w/2}}{2}. \quad (17)$$

The term accounting for repeated sampled inputs satisfies

$$\frac{q(q-1)}{2D} < \frac{q^2}{2D} = \frac{2^{3w}}{2 \cdot 2^{4w}} = 2^{-w-1}.$$

Consequently the probability P_{coll} of obtaining equal digests for two distinct messages obeys

$$P_{\text{coll}} \geq 1 - \exp\left(-\frac{1 - 2^{-3w/2}}{2}\right) - 2^{-w-1} > 0.39 \quad (18)$$

for both $w = 128$ and $w = 256$. The bound approaches $1 - e^{-1/2} \approx 0.393469$ as w increases. The choice $D = 2^{4w}$ gives $q^2/D = 2^{-w}$, which bounds the probability of repeated sampled inputs by the small correction above.

Algorithm 1, applied to $\mathcal{D} = \{0, 1\}^{4w}$ with $q = 2^{3w/2}$, therefore finds a collision with probability greater than 0.39, using at most q hash evaluations and q stored records. Substituting the two word widths gives:

Digest (bits)	Hash evaluations	Stored records
512	2^{192}	2^{192}
1024	2^{384}	2^{384}

Each evaluation processes $4w$ message bits. Writing $n = 4w$ for the digest length, the attack gives an upper bound of $O(2^{3n/8})$ hash evaluations and stored records, compared with the birthday cost of $O(2^{n/2})$ evaluations for an ideal n -bit hash. The corresponding exponents decrease from 256 to 192 for the 512-bit digest and from 512 to 384 for the 1024-bit digest.

Reading the determinant constraint from a digest. Decode the digest into four field elements and apply inv_0 to each entry. This recovers $H_{\text{mat}}(M)$, whose determinant is $2^L \bmod p$. Since this decoding is bijective, it preserves the number of possible values.

7 QSH: Free-Start Collisions and Semi-Free-Start Bounds

QSH admits explicit free-start collisions because its compression function is invertible with respect to the incoming state for every message block and flag. Its complete core permutation also preserves two periodic word spaces. The exact classifications below give a bound on the number of complete CVs and a semi-free-start collision search in the complete tree mode, with one chosen reset shared by both messages and all nodes.

7.1 The core permutation and its exact word structure

The state comprises 64 words of width $w \in \{32, 64\}$, indexed by

$$i = 16x_0 + 4x_2 + x_1, \quad x_0, x_1, x_2 \in \mathbb{Z}_4.$$

We denote its word at a coordinate by $A[x_0, x_1, x_2]$. One round consists of three layers of disjoint, ordered calls to the same four-word transformation G [6, Secs. 3.1–3.4, pp. 7–12]. Their operand coordinates, in order $t = 0, 1, 2, 3$, are

$$\begin{aligned} C &: (t, x_1, x_2) && \text{for fixed } x_1, x_2, \\ D_{01} &: (t, p + t, x_2) && \text{for fixed } p, x_2, \\ D_{12} &: (x_0, t, p + t) && \text{for fixed } x_0, p. \end{aligned} \tag{19}$$

The diagonal index p belongs to $\mathbb{Z}_4$, and all coordinate additions are modulo 4. These groups agree with Tables 15–17 in Appendix A of the specification. In particular, D_{01} includes $(A_0, A_{17}, A_{34}, A_{51})$, while D_{12} includes $(A_0, A_5, A_{10}, A_{15})$.

The transformation G acts on (a, b, c, d) by the following eight updates, executed from left to right within each row and from top to bottom. All additions are modulo 2^w :

$$\begin{aligned} a &\leftarrow a + \text{rotr}_w(b, 1), & d &\leftarrow \text{rotr}_w(d \oplus a, w/2), \\ c &\leftarrow c + \text{rotr}_w(d, w/2 - 1), & b &\leftarrow \text{rotr}_w(b \oplus c, w/2 - 4), \\ a &\leftarrow a + b, & d &\leftarrow \text{rotr}_w(d \oplus a, w/4), \\ c &\leftarrow c + \text{rotr}_w(d, w/4), & b &\leftarrow \text{rotr}_w(b \oplus c, w/4 - 1). \end{aligned} \tag{20}$$

Each call writes its four outputs back to the same ordered group.

The complete permutation is

$$E_w = C \circ (D_{12} \circ D_{01} \circ C)^{n_w}, \quad n_{32} = 9, \quad n_{64} = 18.$$

The specification counts the final application of C as an additional round, reporting 10 and 19. The expression above therefore includes all prescribed layers. Every group in every round applies the same G , with no additional constant injection. Each update in G is a reversible word addition, xor, or rotation. For example, $x \leftarrow x + \text{rotr}_w(y, s) \pmod{2^w}$ is undone by subtraction with the unchanged y , and $x \leftarrow \text{rotr}_w(x \oplus y, s)$ is undone by $x \leftarrow \text{rotr}_w(x, s) \oplus y$. Undoing the updates in reverse order inverts G . Since the calls within each layer use disjoint word groups, each layer is a product of bijections; their composition E_w is therefore also a bijection.

Define the permutation T on word positions by

$$(TA)[x_0, x_1, x_2] = A[x_0, x_1, x_2 + 1].$$

Since the third coordinate is taken modulo four, T has order four.

Proposition 6 (Commuting translation). *For every state A and both specified word widths,*

$$E_w(TA) = T(E_w(A)).$$

Proof. For C and D_{01} , translation replaces the group index x_2 by $x_2 + 1$; for D_{12} , it replaces p by $p + 1$. Every input retains its position t in the ordered call to G . For example, if G_t denotes output word t , then

$$\begin{aligned} (D_{12}(TA))[x_0, t, p + t] &= G_t((A[x_0, s, p + 1 + s])_{s=0}^3) \\ &= (T(D_{12}A))[x_0, t, p + t]. \end{aligned}$$

Thus every layer commutes with T , as does its composition with the other layers, including the final column layer.

Write $\mathcal{A}_w = \mathbb{Z}/2^w\mathbb{Z}$, and put $V_r = \text{Fix}(T^r) \subseteq \mathcal{A}_w^{64}$ for $r = 1, 2$. These are binary linear spaces because T is a linear coordinate permutation; E_w itself is nonlinear.

Proposition 7 (Periodic spaces and membership). *For $r = 1, 2$, $E_w(V_r) = V_r$ and $\dim_{\mathbb{F}_2} V_r = 16rw$. For a fixed input in V_r , its image under a uniformly random permutation belongs to V_r with probability $2^{-(64-16r)w}$.*

Proof. Commutation gives $E_w(V_r) \subseteq V_r$; bijectivity and finiteness give equality. Each of the $16r$ cycles of T^r contributes one freely assignable word. A random permutation sends a fixed input uniformly to the 2^{64w} possible outputs.

For $r = 1$, the input $A[x_0, x_1, x_2] = 4x_0 + x_1$ belongs to V_1 . Membership is tested by comparing each of the three other x_2 positions to position zero for every (x_0, x_1) , using 48 word comparisons. The acceptance probabilities are 1 and 2^{-48w} . Since E_w is public, one could instead compute its expected output and compare it with the response. The structural content here is the preserved space and its inexpensive membership test.

For a partition $\mathcal{P}$ of the 64 coordinates, let $W_{\mathcal{P}}$ consist of states whose words agree within each partition block, with one arbitrary word chosen independently per block. We call $W_{\mathcal{P}}$ the *equality space* associated with $\mathcal{P}$.

Proposition 8 (Exact classifications). *For both prescribed full cores and any coordinate partitions $\mathcal{P}, \mathcal{Q}$,*

$$E_w(W_{\mathcal{P}}) = W_{\mathcal{Q}} \iff W_{\mathcal{P}} = W_{\mathcal{Q}} \in \{V_1, V_2, \mathcal{A}_w^{64}\}.$$

For pure permutations P, Q of the 64 word positions,

$$E_w P = Q E_w \iff P = Q = T^k \text{ for some } k \in \mathbb{Z}_4.$$

Appendix D.1 proves the classifications from explicit finite partition and multiset certificates. Their scope is spaces defined solely by equalities between entire words and pure permutations of word positions.

7.2 A small CV image at one common state

Write $H = (L, R) \in \mathcal{A}_w^{32} \times \mathcal{A}_w^{32}$, and let $M \in \mathcal{A}_w^{32}$ be a message block. The map J_f XORs flag f into word zero of the left half. The specified compression is [6, Secs. 4–5]

$$S = (J_f L, R + M), \quad Y = E_w(S), \quad F_f(H, M) = (Y_L + M, Y_R). \quad (21)$$

All additions and subtractions in this section are wordwise modulo 2^w . The complete CV, denoted $\text{CV}(Z)$, is the first 16 words of Z . State, block, CV,

and final digest widths are respectively $h = 64w, m = 32w, c = 16w, n$. For $(n, w) = (512, 32), (768, 64), (1024, 64)$, the CV widths are 512, 1024, and 1024 bits. QSH-768 truncates to 12 words only at the final digest. QSH words and the length field use little-endian encoding.

Let U_r be the states fixed by the restriction of T^r to a 32-word half, so $V_r = U_r \times U_r$ and $|U_r| = 2^{8rw}$. Equality on coordinate orbits makes U_r closed under wordwise addition and subtraction as well as XOR.

Theorem 1 (CVs obtained from a common state). *Fix one state $H = (L, R)$, one flag f , and $r \in \{1, 2\}$ such that $J_f L \in U_r$. For $B \in U_r$, the messages $M_B = B - R$ are distinct and satisfy*

$$D = 2^{8rw}, \quad |\text{Im}(B \mapsto \text{CV}(F_f(H, M_B)))| \leq K = 2^{4rw}. \quad (22)$$

Proof. The core input $(J_f L, B)$ lies in V_r , so its output Y lies there too. For $Z = F_f(H, M_B)$,

$$Z_L + R = Y_L + B \in U_r.$$

The first 16 words of this normalized vector contain $4r$ independent words. Subtracting the same fixed R preserves the resulting image bound. Translation by $-R$ is injective.

Applying Lemma 1 with $D = K^2$ and $q = \sqrt{K} = 2^{2rw}$ gives a collision between distinct messages with probability greater than 0.39, using independent uniform samples from U_r . For $r = 1$, the budgets are $2^{64}, 2^{128}, 2^{128}$ compression evaluations for the three variants. For $r = 2$, they are $2^{128}, 2^{256}, 2^{256}$. Each search stores at most q records. A record can retain the $8r$ free parameter words and $4r$ normalized CV words, or $12rw$ payload bits, plus indexing overhead. Our propagation argument uses equality of the complete CV at a chunk boundary. Subsequent blocks within a chunk consume the full 64-word state.

7.3 Semi-free-start collisions in the complete tree mode

For $I \in \mathcal{A}_w^{64}$, define QSH_I by replacing every leaf and parent reset with the same I , selected once before the search. All other rules remain as specified. This is a complete 64-word effective reset, not merely a choice of the scalar in the initialization routine. The prescribed API fixes that reset.

The flags START, END, PARENT, and ROOT are 1, 2, 4, and 8, combined by bitwise OR. Padding appends one bit, zeros, and the raw message length ℓ , encoded in 128 bits in little-endian order, giving $N = \lfloor \ell/m \rfloor + 2$ blocks. A chunk contains at most 16 blocks. A full non-root data chunk uses flags $(1, 0^{14}, 2)$. A parent concatenates its two complete CVs into one block and uses flag 4, or 12 at the root. The tree splits at the largest power of two strictly smaller than its leaf count. If there is just one chunk, ROOT is added to its last flag. Lengths are restricted to those representable by the specified encoding and implementation arithmetic.

For each message block M and flag f , define $F_{f,M}(H) = F_f(H, M)$. This map is a permutation of the 64-word state space, with inverse

$$S = E_w^{-1}(Z_L - M, Z_R), \quad F_{f,M}^{-1}(Z) = (J_f S_L, S_R - M). \quad (23)$$

Subtracting M removes feed-forward, after which the core inverse and the inverse injection recover H . For a fixed public 15-block prefix P , define

$$\Psi_P = F_{0,P_{15}} \circ \cdots \circ F_{0,P_2} \circ F_{1,P_1}. \quad (24)$$

Choose $B_0 \in U_r$ and any R , set $H_{15} = (J_2 B_0, R)$, and compute $I = \Psi_P^{-1}(H_{15})$. Computing this reset requires 15 evaluations of E_w^{-1} and fixes one reset for all candidates and nodes.

Theorem 2 (Semi-free-start collision bound for the tree mode). *For any such prefix and either $r = 1, 2$, the chosen I supports $D = 2^{8rw}$ distinct raw messages of exactly 16 blocks, with at most $K = 2^{4rw}$ complete digests under QSH_I. A search using 2^{2rw} compression evaluations finds a collision with probability greater than 0.39. After success, three further compressions compute the common digest.*

Proof. For $B \in U_r$, use the raw message $C_B = P_1 \parallel \cdots \parallel P_{15} \parallel (B - R)$. Its first chunk ends with flag 2 and has the CV image bound of Theorem 1. Padding extends 16 raw blocks to 18 blocks. The last two form a separate, identical padding chunk, reset to I and processed with flags (1, 2). Its CV is therefore common to all candidates. The root, also reset to I , uses flag 12 and the concatenation of the two complete CVs. Its digest is a deterministic function of the first CV.

The state H_{15} is known and reused for all candidates. Each candidate therefore requires one compression evaluation on the final block of the first chunk. Lemma 1 applies to the first CV; equal CVs of distinct sampled messages give equal complete digests. Two padding compressions and one root compression then suffice.

The raw messages have 2048 bytes for QSH-512 and 4096 bytes for QSH-768/1024. For $r = 1$, preparation requires 15 evaluations of E_w^{-1} , the search uses $2^{64}, 2^{128}, 2^{128}$ compression evaluations, and three further compressions compute the common digest. Independent forward verification of both returned messages incurs a further constant cost. QSH-768 uses all 16 CV words at the parent; no intermediate digest truncation is used.

7.4 Explicit free-start collisions

For arbitrary distinct blocks M_0, M_1 , a common flag f , and a prescribed complete output Z , set

$$H_i = F_{f,M_i}^{-1}(Z), \quad i = 0, 1.$$

Then $F_f(H_0, M_0) = F_f(H_1, M_1) = Z$, a free-start compression collision obtained with two evaluations of E_w^{-1} by (23). The input pairs are distinct because $M_0 \neq M_1$. For any padded message fitting in one chunk, invert its compression

calls in reverse order from a selected full final state, using its padded blocks and prescribed flags, including ROOT on the last block. This gives free-start collisions for the complete mode, including for messages with different raw lengths. Appendix D.4 specifies concrete pairs for all three variants and prints a complete QSH-512 witness. This construction follows from invertibility and does not require the periodic spaces.

The construction permits a separate complete reset for each message and uses the inverse permutation E_w^{-1} . If the interface permits selecting the entire 64-word input $A_i \in \mathcal{A}_w^{64}$ to the initialization permutation, the required reset I_i is obtained by choosing $A_i = E_w^{-1}(I_i)$. This requires one additional evaluation of E_w^{-1} per message.

7.5 Prescribed initialization: mode reuse and legal entry

The prescribed reset is $H^0 = E_w(c_0, 0, \dots, 0)$, with

$$(n, w, c_0) \in \{(512, 32, 17), (768, 64, 1), (1024, 64, 7)\}.$$

Evaluation gives the following words:

$n \ H_1^0$	H_5^0	H_9^0
512 28746253	95566f10	273fe3ed
768 cd2aec670a91dfe f02bc7e19f0b6962	6a466260d2f99e59	
1024 3c1836f1bf4fe859 527128a1cb34cbc9	15b65338129764f3	

The first flag changes only word zero; the first message changes only the right half. Thus $H_1^0 \neq H_5^0$ excludes entry into V_1 at the first compression call, and $H_1^0 \neq H_9^0$ excludes entry into V_2 at that call. These facts leave later entry through a legal prefix open.

Full data chunks use identical resets and flags, with no input specifying the chunk position. This gives a separate mode consequence.

Corollary 4 (Multicollisions under the prescribed initialization). *Let $c = 16w$. With probability greater than 0.39, a search using $2^{c/2} + 19$ compressions constructs four distinct 32-block raw messages with the same QSH digest under the prescribed initialization. The count assumes the prescribed reset is cached, includes their common digest, and uses at most $2^{c/2}$ search records.*

Fix a 15-block prefix from H^0 , and vary the unrestricted last block. Its 2^{32w} inputs have at most 2^{16w} CVs, so Lemma 1 gives two distinct chunks C_0, C_1 with one CV. The four messages $C_0C_0, C_0C_1, C_1C_0, C_1C_1$ have identical sequences of leaf CVs after padding. Appendix D.3 proves propagation and the count, including the general 2^t -message family. This uses established collision composition and tree substitution principles [9,1]. The specification gives the estimate $t2^{c/2}$ for a 2^t -collision using Joux’s multicollision construction [6, Sec. 7.5]. Our construction searches for one pair of chunks with equal complete CVs and reuses that pair at every chunk position. The CV search exponents are 256, 512, and 512; the ordinary digest birthday exponents are 256, 384, and 512, respectively.

The smaller periodic family requires a legal prefix from H^0 whose verified endpoint satisfies $H_{15} = (L, R)$ and $J_2 L \in U_r$. If an entry procedure succeeds with probability p_{entry} using at most C_{entry} forward compressions, including endpoint verification, fresh sampling then gives

$$\text{cost} \leq C_{\text{entry}} + 2^{2rw} + 3, \quad \Pr[\text{collision}] \geq 0.39p_{\text{entry}}. \quad (25)$$

Conditioning on each valid endpoint applies Theorem 1 and the padding argument of Theorem 2 with $I = H^0$. Other entry work and storage remain additional costs. Reverse evaluation of a prefix selects another reset and does not reach the required state from H^0 . The search bounds are theoretical: the concrete free-start examples use separate resets. No collision pair has been computed in the semi-free-start setting or under the prescribed initialization.

8 WChain: Invariant Sets and Short Cycles in Message Expansion

WChain’s message expansion preserves sets of words fixed by bit rotation. When each word is either zero or $2^{64} - 1$, the expansion is determined by 18 bits in Version I and 36 bits in Version II. Within these sets, we construct sequences of expanded blocks with periods one, three, and six. The sequences supply the expanded blocks for all 18 rounds in Version I and all 24 rounds in Version II, as well as the block used for final whitening. We also construct messages whose padded blocks and checksum blocks belong to these sets.

8.1 The expansion restricted to words fixed by rotation

The parameter table specifies state sizes $b = 576, 1152$, message block sizes $\mu = 2b$, and round counts $R = 18, 24$, respectively [8, Table 1, p. 6]. Write $n = b/64$, so that $n = 9$ or 18 . A message block supplies two consecutive b -bit seeds K_0, K_1 , parsed as big-endian words. On each nine-word half, the expansion uses

$$D(A)_i = \text{rotl}_{64}(A_i, \rho_i), \quad \rho = (0, 1, 10, 20, 25, 32, 41, 46, 50), \quad (26)$$

$$\text{MEF}(A) = S(D(A)), \quad K_j = \text{MEF}(K_{j-1}) \oplus K_{j-2}, \quad 2 \leq j \leq R. \quad (27)$$

In Version II, MEF acts independently on the two nine-word halves [8, Section 3.5, pp. 13–14]. The S-box acts at each bit position, with indices modulo nine:

$$S_9(a)_i = a_i \oplus \overline{(a_{i+1} \wedge a_{i+2})}. \quad (28)$$

Both negations are present in the original equation [8, Eq. (22), p. 11]. The data rounds use $K_0, \dots, K_{R-1}$; K_R supplies final whitening [8, Eqs. (10)–(11), p. 9]. Thus there are $R + 1$ expanded blocks but only $R - 1$ MEF evaluations after initialization.

Let $J = 2^{64} - 1$, and define the linear embedding

$$\phi : \mathbb{F}_2^9 \longrightarrow (\mathbb{F}_2^{64})^9, \quad \phi(a)_i = Ja_i.$$

Here i is a word index, and Ja_i equals zero for $a_i = 0$ and J for $a_i = 1$. We identify a nine-word array with its concatenation in $\mathbb{F}_2^{576}$. The image of ϕ therefore consists of arrays in which each word is independently zero or J . Rotations act trivially on these words, and the 64 S-box columns receive identical inputs. Consequently,

$$\text{MEF}(\phi(a)) = \phi(S_9(a)), \quad \phi(a) \oplus \phi(b) = \phi(a \oplus b). \quad (29)$$

Proposition 9 (Invariance under expansion). *For $d \mid 64$, let*

$$U_d^{(n)} = \{A \in (\mathbb{F}_2^{64})^n : \text{rotl}_{64}(A_i, d) = A_i \text{ for every } i\}.$$

This is an nd -dimensional linear space. The map on consecutive blocks

$$\mathcal{T}(A, B) = (B, A \oplus \text{MEF}(B))$$

permutes $U_d^{(n)} \times U_d^{(n)}$. For $d = 1$, its restriction is conjugate to one copy (Version I) or two independent copies (Version II) of the 18-bit permutation

$$T(a, b) = (b, a \oplus S_9(b)). \quad (30)$$

Proof. A word fixed by rotation through d positions is determined by d bits: because $d \mid 64$, its bits with indices congruent modulo d must coincide, and the d residue classes are independent. Each of the n words therefore contributes d free bits, giving dimension nd and cardinality 2^{nd} .

Rotations commute, so rotating a word fixed by $\text{rotl}_{64}(\cdot, d)$ produces another fixed word. Bitwise Boolean functions, including complement, preserve equality of corresponding positions separated by d . Thus both D and S , and hence MEF , preserve $U_d^{(n)}$. Xor closure then proves that $\mathcal{T}$ maps the pair space into itself.

Given $(C, D) = \mathcal{T}(A, B)$, its first component says $B = C$, and its second says $A = D \oplus \text{MEF}(C)$. Hence

$$\mathcal{T}^{-1}(C, D) = (D \oplus \text{MEF}(C), C).$$

This inverse preserves $U_d^{(n)} \times U_d^{(n)}$, so the restriction of $\mathcal{T}$ to this set is a permutation.

For $d = 1$, the embedding ϕ is a bijection from nine bits to the nine-word arrays just defined. On pairs, write $\Phi(a, b) = (\phi(a), \phi(b))$. Equation (29) gives

$$\mathcal{T}(\Phi(a, b)) = (\phi(b), \phi(a \oplus S_9(b))) = \Phi(T(a, b)).$$

This proves the conjugacy. Each half of Version II obeys this identity independently, giving the direct product of two 18-bit maps. $\square$

The selected message sets $U_1^{(n)} \times U_1^{(n)}$ therefore have dimensions 18 and 36, within the 1152- and 2304-bit message spaces. More generally, applying the same rotation to every word of both seeds rotates every expanded block by that amount: the rotations in (26) commute, and the S-box is bitwise.

Table 2. Shortest cycles of the permutation T . Each cycle contains as many seed pairs as its length. Enumeration covers all 2^{18} states; Table 3 gives all cycle lengths.

Cycle length	1	3	6	9	12	14	18	24
Cycle count	1	1	450	9	414	3	9	135
Seed pairs	1	3	2700	81	4968	42	162	3240

8.2 Fixed points and cycle lengths

Equation (28) gives $S(0^b) = 1^b$ and $S(1^b) = 0^b$. Hence the seed pair $(1^b, 1^b)$ produces $K_j = 1^b$ for every j , whereas $(0^b, 0^b)$ produces

$$0^b, 0^b, 1^b, 0^b, 0^b, 1^b, \dots$$

The two sequences have least periods one and three, respectively. For the zero seed pair, the first three updates are explicitly $K_2 = S(0^b) \oplus 0^b = 1^b$, $K_3 = S(1^b) \oplus 0^b = 0^b$, and $K_4 = S(0^b) \oplus 1^b = 0^b$. Thus the pair (K_3, K_4) returns to (K_0, K_1) after three steps, while neither of the first two pair states equals the starting pair.

An exhaustive decomposition of the 2^{18} states of T in (30) gives 1,336 cycles and maximum period 3,009. These counts describe Version I and each half of Version II. Table 2 lists the shortest cycle lengths; Appendix C gives the enumeration algorithm, its correctness proof, and the complete table. The sequence $(K_j)_{j \geq 0}$ has the same period as the orbit of (K_0, K_1) : a period of the sequence repeats every pair (K_j, K_{j+1}) , and a repeated pair determines all subsequent blocks by the recurrence. If c_d counts cycles of length d , each such cycle contains d different seed pairs. Period dividing six therefore selects exactly $\sum_{d|6} dc_d$ pairs. Enumeration gives $c_1 = 1$, $c_2 = 0$, $c_3 = 1$, and $c_6 = 450$, so this count is

$$1 + 3 + 6 \cdot 450 = 2704$$

seed pairs. Version II has two independent orbits, whose joint period is their least common multiple. It therefore has exactly $2704^2 = 7,311,616$ message blocks in $U_1^{(18)} \times U_1^{(18)}$ with period dividing six: $\text{lcm}(d_1, d_2) \mid 6$ holds exactly when both $d_1 \mid 6$ and $d_2 \mid 6$, and the choices of seeds for the two halves are independent.

For a nonconstant example, interpret mask bit i as the coefficient of word i in ϕ . The seeds $a = 0\text{ef}$, $b = 177$ give the repeating mask sequence

$$0\text{ef}, 177, 023, 023, 177, 0\text{ef}. \quad (31)$$

The six pairs (K_j, K_{j+1}) are distinct, establishing least period six. Their encodings are, in order,

$$\begin{aligned} &(0\text{ef}, 177), (177, 023), (023, 023), \\ &(023, 177), (177, 0\text{ef}), (0\text{ef}, 0\text{ef}). \end{aligned}$$

Applying (30) to the last pair returns the first. Applying ϕ to the sequence gives all 19 expanded blocks of Version I. Using the same seeds in both halves gives all 25 blocks of Version II. Allowing each half to use any of the 2704 pairs with period dividing six gives the 2704^2 blocks counted above.

8.3 Padding and checksum blocks

The padding appends a one bit and the minimum number of zero bits needed to reach a multiple of μ [8, Section 3.1, p. 7]. Thus the legal message $M = 1^{\mu-1}$ becomes the block 1^μ , producing a constant expansion in both versions. The last word of $\phi(177)$ is J . Serialize the seeds of (31) as a message block, using them in both halves for Version II. Removing its final bit yields a legal $(\mu-1)$ -bit message; padding restores that bit as its delimiter. The block 0^μ is reachable as a nonfinal block. The counts above enumerate compression inputs; the constructions here specify how such blocks occur in padded messages.

The final checksum is $\Sigma = M_1 \oplus \cdots \oplus M_\ell$ [8, Eq. (5), p. 8]. If each word in every padded block is zero or J , each word in Σ is also zero or J , because these two values are closed under xor. For every supported $\ell \geq 1$, the message $1^{\ell\mu-1}$ pads to ℓ copies of 1^μ . Every ordinary compression call then uses the fixed expansion; the checksum call uses the fixed expansion for odd ℓ and an expansion of period three for even ℓ . This follows coordinatewise: xoring ℓ copies of J gives J when ℓ is odd and zero when ℓ is even. Thus the checksum's two seeds are either $(1^b, 1^b)$ or $(0^b, 0^b)$, whose expansions were derived above. For arbitrary padded blocks in the invariant set, xor closure places the checksum in the same set. Its period is determined by the orbit of its own seeds under $\mathcal{T}$.

9 Cuishen: Short Cycles in Message Expansion

Cuishen's message expansion preserves 256 values of its 1024-bit register throughout all 64 rounds. On this set, each update cyclically shifts eight binary coordinates by three positions. For two key schedules initialized with the same chaining register and counter, the XOR differences between their round keys have periods dividing eight.

9.1 An eight-bit recurrence

Cuishen-512 uses Octarx-256, whereas Cuishen-768 and Cuishen-1024 use Octarx-512. Both ciphers have 64 rounds and the same message update [15, Algorithms 4–5, PDF pp. 13–14]. Write the message register as $m = (m_0, \dots, m_{15})$, with 64-bit words, and let $J = 2^{64} - 1$. Consider the eight-dimensional linear space

$$\mathcal{U} = \{m : m_{2j} = m_{2j+1} = \alpha_j J, \quad \alpha_j \in \mathbb{F}_2, \quad 0 \leq j < 8\}.$$

The only additions in the update occur in four output words:

$$\begin{aligned} m'_2 &= \text{rotl}_{64}(m_8, 29) \boxplus \text{rotl}_{64}(m_{10} \oplus m_{11}, 6), \\ m'_3 &= \text{rotl}_{64}(m_9, 63) \boxplus \text{rotl}_{64}(m_{10} \oplus m_{11}, 12), \\ m'_{10} &= \text{rotl}_{64}(m_0, 14) \boxplus \text{rotl}_{64}(m_2 \oplus m_3, 19), \\ m'_{11} &= \text{rotl}_{64}(m_1, 43) \boxplus \text{rotl}_{64}(m_2 \oplus m_3, 28), \end{aligned}$$

where $\boxplus$ denotes addition modulo 2^{64} . The other outputs copy input words. Specifically, for $i \notin \{2, 3, 10, 11\}$, $m'_i = m_{(i+6) \bmod 16}$. The four displayed equations replace those four coordinates of the same cyclic shift by six word positions.

Proposition 10 (Message expansion on the selected set). *The message update permutes $\mathcal{U}$ and induces*

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7) \mapsto (\alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_0, \alpha_1, \alpha_2). \quad (32)$$

The numbers of members with least periods 1, 2, 4, 8 are respectively 2, 2, 12, 240, giving 36 cycles.

Proof. For a member of $\mathcal{U}$, both $m_{10} \oplus m_{11}$ and $m_2 \oplus m_3$ equal zero. Their rotations also equal zero, so the second summand of every displayed addition vanishes. The first summand is a rotation of zero or J , which leaves that word unchanged. Consequently $m'_i = m_{(i+6) \bmod 16}$ holds at all sixteen positions. Grouping the outputs into pairs gives

$$(m'_{2j}, m'_{2j+1}) = (\alpha_{(j+3) \bmod 8} J, \alpha_{(j+3) \bmod 8} J),$$

which proves (32) and closure. The inverse cyclic shift is by minus three, so the update permutes the $2^8 = 256$ members. Induction gives $\alpha_j^{(r)} = \alpha_{(j+3r) \bmod 8}^{(0)}$ at every step.

After eight steps the shift is zero modulo eight. Every least period therefore divides eight. For $p \mid 8$, a vector fixed after p steps satisfies $\alpha_j = \alpha_{(j+3p) \bmod 8}$. Addition by $3p$ modulo eight partitions the positions into $\gcd(8, 3p) = p$ orbits. A fixed vector chooses one independent bit on each orbit, hence there are 2^p such vectors. Writing N_p for the number of vectors of least period p , subtract the cases with smaller periods:

$$\begin{aligned} N_1 &= 2, & N_2 &= 4 - N_1 = 2, \\ N_4 &= 16 - N_1 - N_2 = 12, & N_8 &= 256 - N_1 - N_2 - N_4 = 240. \end{aligned}$$

Each period- p cycle has p members, giving $N_1/1 + N_2/2 + N_4/4 + N_8/8 = 2 + 1 + 3 + 30 = 36$ cycles. The induction applies to all iterates, and thus to all 64 prescribed rounds. $\square$

9.2 XOR differences between round keys

Let $h = 4$ for Octarx-256 and $h = 8$ for Octarx-512, so the cipher block size is $n = 64h$ bits. A master key is parsed as $m^{(0)} \parallel B^{(0)} \parallel \text{Cnt}^{(0)}$, with a 1024-bit message register, an h -word chaining register, and a 128-bit counter. In normal compression, the key for block m_j is $m_j \parallel b_{j-1} \parallel \text{Cnt}_j$, where $H_{j-1} = t_{j-1} \parallel b_{j-1}$ is the incoming chaining value. Thus $m^{(0)} = m_j$ and $B^{(0)} = b_{j-1}$. Finalization instead uses $H_N \parallel m_{\text{final}} \parallel \text{Cnt}_{\text{final}}$; the message register then contains the first 1024 bits of this key [15, Sections 3.4–3.5].

Write $\alpha = \alpha^{(0)}$ for the initial eight-bit vector. Subscripts of α , and of its XOR differences below, are reduced modulo eight. Before update r , the round key selects message words at positions 0, 1, 8, 9. Since $\alpha_j^{(r)} = \alpha_{j+3r}^{(0)}$, these words are

$$(\alpha_{3r}, \alpha_{3r}, \alpha_{3r+4}, \alpha_{3r+4})J, \quad \text{indices modulo eight.} \quad (33)$$

Denote their contribution to the round key by $E_r(\alpha)$, appending four zero words when $h = 8$.

The remaining registers evolve as follows:

$$\begin{aligned} B^{(r+1)} &= (B_1^{(r)}, \dots, B_{h-1}^{(r)}, \text{rotl}_{64}(B_0^{(r)}, \rho_h) \oplus B_1^{(r)}), \\ \text{Cnt}^{(r+1)} &= (\text{Cnt}_1^{(r)}, \text{rotl}_{64}(\text{Cnt}_0^{(r)}, 2)), \quad \rho_4 = 27, \quad \rho_8 = 47. \end{aligned} \quad (34)$$

Within one execution of the key schedule, these recurrences depend only on $B^{(0)}$ and $\text{Cnt}^{(0)}$. Fix these initial registers and define

$$R_r = B^{(r)} \oplus (\text{Cnt}_0^{(r)}, \text{Cnt}_1^{(r)}, 0, \dots, 0).$$

The round key for the message register determined by α is

$$W_r(\alpha) = R_r \oplus E_r(\alpha).$$

For two initial vectors $\alpha, \tilde{\alpha}$ and the same $B^{(0)}, \text{Cnt}^{(0)}$, the terms R_r cancel:

$$\Delta W_r = W_r(\alpha) \oplus W_r(\tilde{\alpha}) = E_r(\alpha) \oplus E_r(\tilde{\alpha}).$$

Writing $\delta = \alpha \oplus \tilde{\alpha}$, we obtain

$$\Delta W_r = (\delta_{3r}, \delta_{3r}, \delta_{3r+4}, \delta_{3r+4})J$$

for Octarx-256, with four zero words appended for Octarx-512. Since $3(r+8) \equiv 3r \pmod{8}$,

$$\Delta W_{r+8} = \Delta W_r, \quad 0 \leq r < 56. \quad (35)$$

The first four message selections read the bit pairs $(0, 4), (3, 7), (6, 2), (1, 5)$. They cover all eight bits, so the sequences $E_r(\alpha)$ are distinct for all 256 register values: equality of the first four tuples forces equality of every bit. For fixed initial registers, equality of two sequences of round keys would give equality of their E_r terms by cancelling R_r . Hence all 256 register values produce distinct sequences of round keys.

9.3 Contribution of the chaining value and counter

Since $E_{r+8}(\alpha) = E_r(\alpha)$, for $0 \leq r < 56$ we have

$$W_{r+8}(\alpha) \oplus W_r(\alpha) = R_{r+8} \oplus R_r.$$

Thus $W_{r+8}(\alpha) = W_r(\alpha)$ if and only if $R_{r+8} = R_r$.

Consider the first compression of a 1024-bit message, for which $B^{(0)}$ is the lower half of the prescribed IV and $\text{Cnt}^{(0)} = (0, 1024)$. All 256 blocks in $\mathcal{U}$ can occur in this invocation with the same initial registers, so (35) applies to them directly. For the zero block, $E_r(0) = 0$, and two counter updates give

$$\text{Cnt}^{(2j)} = (0, \text{rotl}_{64}(1024, 2j)).$$

The first counter word is therefore zero in rounds 0 and 8, giving $(W_0)_0 = B_0^{(0)}$ and $(W_8)_0 = B_0^{(8)}$. For $h = 4$, four applications of (34) give

$$\begin{aligned} B_0^{(4)} &= \text{rotl}_{64}(B_0^{(0)}, 27) \oplus B_1^{(0)}, \\ B_1^{(4)} &= \text{rotl}_{64}(B_1^{(0)}, 27) \oplus B_2^{(0)}. \end{aligned}$$

Applying the first identity again, rotation distributes over XOR and the two copies of $\text{rotl}_{64}(B_1^{(0)}, 27)$ cancel. For $h = 8$, eight updates move the first appended word to position zero. Hence

$$B_0^{(8)} = \begin{cases} \text{rotl}_{64}(B_0^{(0)}, 54) \oplus B_2^{(0)}, & h = 4, \\ \text{rotl}_{64}(B_0^{(0)}, 47) \oplus B_1^{(0)}, & h = 8. \end{cases}$$

The IV words needed for this calculation and the resulting key word are given below. The third column is $B_2^{(0)}$ for Cuishen-512 and $B_1^{(0)}$ for the other two variants.

Digest (bits)	$B_0^{(0)} = (W_0)_0$	$B_2^{(0)}$ or $B_1^{(0)}$	$(W_8)_0$
512	44f9363580e83d02	b467369e08efd70e	f4f608d3858fed01
768	869d6342f6d22822	11076689f6aff6b0	051625c7470e8dd9
1024	f281f2397b1d4610	77c9c2114e14fd92	d4c1bb51b708401c

Thus $W_0 \neq W_8$ in this first compression for every variant.

The cipher rounds also inject constants c_r . In particular, the specified values in rounds 0 and 8 are [15, Section 4.5]

	c_0	c_8
Octarx-256	243f6a8885a308d3	9216d5d98979fb1b
Octarx-512	b7e151628aed2a6a	90cfd47d7c19bb42.

At their injection, these constants cancel in the XOR difference between two executions of the same round. Their values differ between rounds 0 and 8. Equation (35) specifies the differences in the injected keys. Establishing a differential trail through the cipher also requires following the state through its modular additions and rotations.

10 Discussion

MoFang and Neulaser admit explicit collisions through different mechanisms. In MoFang, corresponding message differences cancel at every key injection, giving the same compression output for any incoming chaining state. Blocks can therefore be chosen independently throughout a message, and equality is preserved during XOF squeezing. Since each expansion step preserves the cancellation, adding rounds with the same expansion and key injection leaves the construction intact.

Neulaser maps the constructed messages to equal states after one update. Direct loading of the chaining vector and message places the differing word at a

position whose contribution to the next state is determined by its local feedback. Equal feedback then gives equal states, and the remaining rounds, padding, and output operations preserve this equality.

CHIME and CHAMP restrict the number of possible digests on large message sets. Lemma 1 relates these two cardinalities to the probability of a collision between distinct messages. In CHIME, the invariant subspace makes the digest a function of two 64-bit words. In CHAMP, the determinant fixes one relation among the four matrix entries. These arguments give theoretical upper bounds on the number of hash evaluations needed to find a collision; the searches also store at most one record per evaluation. The explicit examples illustrate the algebraic constraints, while the success probabilities follow from the lemma.

For every message block and flag, the QSH compression function is invertible with respect to the incoming state. This gives explicit free-start collisions. The periodic structure yields bounds on the cost of finding semi-free-start collisions in the complete tree mode. At an admissible state, the periodic message family bounds the number of complete leaf CVs. Selecting one common full reset by inverse prefix evaluation places the family at a legal chunk boundary, after which equal CVs give equal final digests. The corollary under the prescribed initialization uses identical chunk maps and one unrestricted CV collision. Its reuse across positions changes the multicollision construction, while retaining the CV birthday exponent. The smaller bound obtained from the periodic family under the prescribed initialization depends on the cost of reaching the required state through a valid message prefix, as specified in (25).

The classifications concern the equality spaces $W_{\mathcal{P}}$ and pure permutations of word positions. Removing a coordinate relation alone need not remove small compression images: message injection, feed-forward, and CV projection remain part of any proposed modification’s evaluation. The free-start examples use separate complete resets. The search budgets in the semi-free-start setting and under the prescribed initialization are theoretical.

WChain and Cuishen exhibit invariant sets in their message expansions. For the constructed WChain messages, padding and the checksum preserve membership in the selected sets; further analysis must relate these expansions to the evolution of the compression state. Cuishen’s periodic XOR differences between round keys hold for legal first blocks under the prescribed IV and a common counter. A hash attack would additionally require a relation between the cipher outputs that survives feed-forward and finalization.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sufficient conditions for sound tree and sequential hashing modes. *International Journal of Information Security* **13**(4), 335–353 (2014). <https://doi.org/10.1007/s10207-013-0220-y>, <https://keccak.team/files/TreeHashing.pdf>
2. Bi, W., Zhao, B., Dong, X., Jia, K., Qin, L.: MoFang hash function. Submission to the ICCS NGCC program (Jun 2026), dated June 29, 2026

3. CHAMP: Cayley Hashing with Matrix Products—algorithm specifications. Submission to the ICCS NGCC program
4. The hash function CHIME. Submission to the ICCS NGCC program (Apr 2026)
5. CHIME-512: Reference and optimized implementations. Software accompanying the ICCS NGCC submission
6. Hou, S., Lin, D., Wang, S.: An efficient tree-structure hash: QuantaSylva Hash (QSH). Submission to the ICCS NGCC program (Jun 2026), dated June 30, 2026
7. Institute of Commercial Cryptography Standards: Announcement on launching the next-generation commercial cryptographic algorithms program (NGCC). Official ICCS announcement (Feb 2025), https://niccs.org.cn/niccs/Notice/pc/content/content_1937428197396713472.html, published February 5, 2025. Accessed September 21, 2026
8. Jia, K., Liu, G., Hou, Q., Peng, Z., Shen, Y., Lin, X., Lu, Z.: WChain hash function. Submission to the ICCS NGCC program
9. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M. (ed.) *Advances in Cryptology – CRYPTO 2004*. Lecture Notes in Computer Science, vol. 3152, pp. 306–316. Springer (2004). https://doi.org/10.1007/978-3-540-28628-8_19
10. Karpman, P., Peyrin, T., Stevens, M.: Practical free-start collision attacks on 76-step SHA-1. In: Gennaro, R., Robshaw, M. (eds.) *Advances in Cryptology – CRYPTO 2015, Part I*. pp. 623–642. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_30
11. Leander, G., Minaud, B., Rønjom, S.: A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In: Oswald, E., Fischlin, M. (eds.) *Advances in Cryptology – EUROCRYPT 2015, Part I*. Lecture Notes in Computer Science, vol. 9056, pp. 254–283. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_11, <https://eprint.iacr.org/2015/068>
12. MoFang: Reference and optimized implementations. Software accompanying the ICCS NGCC submission dated June 29, 2026
13. Neulaser v2: Reference and optimized implementations. Software accompanying the ICCS NGCC submission dated June 30, 2026
14. Qiao, K., Chang, H., Ding, Y., Zhu, L.: Neulaser (v2). Submission to the ICCS NGCC program (Jun 2026), dated June 30, 2026
15. Shen, Y., Cui, T., Liu, Z., Chang, M., He, Y., Li, F., Liu, T.: The hash function Cuishen. Submission to the ICCS NGCC program (Jun 2026), dated June 30, 2026

A Neulaser: Initialization, Global Update, and Output

A.1 Global update and output

This appendix completes the update in (12). All words have 32 bits; additions in the confusion vector are modulo 2^{32} . Module indices are reduced modulo k . For each supported value of k , the four pairs (a_j, b_j) are

k	(a_0, b_0)	(a_1, b_1)	(a_2, b_2)	(a_3, b_3)
3	(0, 1)	(1, 2)	(2, 0)	(1, 0)
4	(0, 1)	(2, 0)	(1, 3)	(3, 2)
5	(0, 1)	(2, 4)	(3, 1)	(4, 3)

For round r and module i , first compute

$$\begin{aligned} x_i &= s_{i+r+a_0,14} \oplus \text{rotl}_{32}(s_{i+r+b_0,9}, 7), \\ y_i &= s_{i+r+a_1,9} + s_{i+r+b_1,6} \pmod{2^{32}}, \\ z_i &= s_{i+r+a_2,6} \oplus \text{rotl}_{32}(s_{i+r+b_2,1}, 11), \\ w_i &= s_{i+r+a_3,1} + s_{i+r+b_3,14} \pmod{2^{32}}. \end{aligned}$$

Then set

$$\Gamma_r^{(i)} = F(x_i, y_i, z_i, w_i) \oplus \text{rotl}_{32}(9\text{e}3779\text{b}9 \oplus r \oplus i, (r+i) \bmod 32),$$

where F is defined in Section 4. A rotation by zero is the identity. Every entry on the right is read from the state before the simultaneous update and uses only positions 14, 9, 6, and 1.

After loading $V\|K$ as specified in Section 4, apply the updates for $r = 0, \dots, 31$ without output. If L_{out} output bits are required, the number of further updates is $N = \lceil L_{\text{out}}/(32k) \rceil$. For $r = 32, \dots, 31 + N$, first update the state, then emit the following word for each $i = 0, \dots, k-1$, in that order:

$$o_i = ((s_{i,13} \oplus s_{i+1,8}) + (s_{i+2,4} \oplus s_{i+3,2})) \bmod 2^{32}.$$

Each o_i is serialized in big-endian order. Concatenate these words over successive rounds and truncate to the required output length. A hash invocation requires at most $2n + 64$ output bits. For $(n, k) = (512, 3), (768, 4), (1024, 5)$, the maximum values of N are respectively 12, 13, and 14, so all required round numbers satisfy $0 \leq r \leq 45$. The integers r and i in the confusion vector are therefore represented exactly as 32-bit words. The output is a deterministic function of the state after the first blank update. Consequently, the state equality in Proposition 2 gives equality of all output bits required by the hash, for any common initial chaining vector.

A.2 Initialization and common digests

The prescribed initial vectors in the v2 implementations [13] are prefixes of one word sequence. For $0 \leq i < (n + 64)/32$, define

$$\begin{aligned} t_i &= 9\text{e}3779\text{b}9(i+1) \bmod 2^{32}, \\ v_i &= \text{rotl}_{32}(t_i, i \bmod 32) \\ &\quad \oplus \text{rotl}_{32}(6\text{a}09\text{e}667 \oplus 1024, (7i) \bmod 32). \end{aligned}$$

Then IV_n is the concatenation, in increasing order of i , of the words v_i serialized in big-endian order. This formula, the update and output equations above, and (13) determine the complete evaluation of the messages in Corollary 3. Their common digests are listed below; each line continues the same hexadecimal byte string.

Neulaser-512.

f899bf50663e6e04fa5374baf28886153ea332bf25443817eb2abbca88072458
dbd7a077712aae99fb112b7672c85a705700e027b348516cc45f8fff537713d3

Neulaser-768.

ce762a05b18ab4fd659e6e601e4236d0908a59ef0feae7dfc71b9580af2f7a5e
3cd9e61caa7f2a59dec94cd7b45dc6f9b700a5260145a7f4531e6a207ffc8930
1f19c342a8a244c17df9140b9e37d9393aa63dd242bc5f9c47a1c8d7edd5094e

Neulaser-1024.

9004dccd0bc7fb735b191dba252363bd283b6e8faa065725e74520c980ed34e2
2a13973901e7eb9bca06e47ced3a9f9dd50130787f4fe09e5014c5a8bb571e07
4174a9a260a81ad126e56fc19e1bd5d8a85c8635c5cc77f97da4290157f45d6d
0bbace9bb06e88e991d1222e811e0e7167a9d3c58512eaa67118ab2740aad8d2

B The CHIME-512 Round Function

Using the layer names in the specification, round r is $RP_r = \tau_r \circ \chi_r \circ AC_r \circ \lambda$, and the permutation is $P = RP_{14} \circ \dots \circ RP_0$. The subscripts record the dependence on the round number. The state is $(A_0, A_1, B_0, B_1, C_0, C_1)$, with each component a 256-bit array. Within an array, words are ordered from most to least significant. A word permutation maps X to Y by $Y_i = X_{j_i}$. The table gives $(j_0, \dots, j_{m-1})$ for each permutation:

permutation	word width	$(j_0, \dots, j_{m-1})$
Π_{perm32}	32	$(3, 0, 1, 2, 7, 4, 5, 6)$
Π_{perm64}	64	$(1, 2, 3, 0)$
Π_{swap64}	64	$(1, 0, 3, 2)$
Π_{swap128}	128	$(1, 0)$.

Let R_t , L_t , and Q_t act independently on the four 64-bit words of an array, respectively by left rotation through t , left shift through t with reduction modulo 2^{64} , and logical right shift through t . These abbreviate the specification's rotation and shift operations; Q_t uses the logical right shift used by both submitted implementations.

The linear layer λ performs the following assignments in the displayed order; each assignment sees all earlier changes:

$$\begin{aligned}
B_1 &\leftarrow B_1 \oplus \Pi_{\text{perm32}}(C_1), \\
B_0 &\leftarrow B_0 \oplus A_1, \\
C_0 &\leftarrow C_0 \oplus R_{29}(A_0), \\
B_1 &\leftarrow B_1 \oplus R_{31}(A_1), \\
C_1 &\leftarrow C_1 \oplus R_{13}(A_0), \\
C_0 &\leftarrow C_0 \oplus B_1, \\
A_0 &\leftarrow \Pi_{\text{perm64}}(A_0), \\
C_1 &\leftarrow C_1 \oplus \Pi_{\text{perm32}}(B_1), \\
A_1 &\leftarrow \Pi_{\text{swap128}}(A_1), \\
A_0 &\leftarrow A_0 \oplus \Pi_{\text{swap64}}(C_0), \\
A_1 &\leftarrow A_1 \oplus \Pi_{\text{perm32}}(B_0), \\
B_1 &\leftarrow B_1 \oplus L_7(A_0), \\
C_1 &\leftarrow C_1 \oplus Q_5(A_1).
\end{aligned} \tag{36}$$

The layer AC_r performs $A_0 \leftarrow A_0 \oplus RC[r]$, where

$$\begin{aligned}
RC[r] &= W(X_r), \\
X_r &= \text{deadbeef00000000} \oplus r \oplus (9\text{e3779b97f4a7c15}(r+1) \bmod 2^{64}).
\end{aligned}$$

Define the bitwise map

$$\chi(X, Y, Z) = (X \oplus (\neg Y \wedge Z), Y \oplus (\neg Z \wedge X), Z \oplus (\neg X \wedge Y)).$$

The three output components use the same input triple. The layer χ_r applies this map to (A_0, B_0, C_0) and (A_1, B_1, C_1) in even rounds, and to (A_0, B_1, C_1) and (A_1, B_0, C_0) in odd rounds. The two triples are disjoint in each case. Finally, τ_r replaces $(A_0, A_1, B_0, B_1, C_0, C_1)$ simultaneously by

$$\begin{cases} (C_1, A_0, A_1, B_0, B_1, C_0), & \lfloor r/3 \rfloor \text{ even,} \\ (A_1, B_0, B_1, C_0, C_1, A_0), & \lfloor r/3 \rfloor \text{ odd.} \end{cases}$$

To evaluate CHIME-512, start with six zero arrays and pad a message M of bit length L as $M \parallel 1 \parallel 0^k \parallel 1$, where $k = (-L - 2) \bmod 512$ and $0 \leq k < 512$. Process the padded blocks in order. Xor a block's first and last 256 bits into A_0 and A_1 , respectively, then perform rounds $r = 0, \dots, 14$. The round index restarts at zero for each block. After the last block, serialize $A_0 \parallel A_1$ in big-endian order. These equations specify the evaluation with big-endian encoding in Section 5.5; all shifts act on individual 64-bit words.

C Cycle Enumeration for WChain

Encode $a = (a_0, \dots, a_8)$ by the integer $\sum_{i=0}^8 a_i 2^i$, and encode a pair (a, b) by $z = 512a + b$. In this encoding, S_9 in (28) is

$$S_9(b) = \sum_{i=0}^8 (b_i \oplus (1 \oplus ((1 \oplus b_{i+1})b_{i+2}))) 2^i, \quad b_i = \lfloor b/2^i \rfloor \bmod 2,$$

where the subscripts $i+1, i+2$ are reduced modulo nine. Thus the integer encoding of $T(a, b)$ is

$$\text{next}(z) = 512(z \bmod 512) + (\lfloor z/512 \rfloor \oplus S_9(z \bmod 512)).$$

Algorithm 2 counts every cycle using this formula.

Algorithm 2 Cycle counts for the reduced WChain permutation

```

1: Set seen[0], ..., seen[218 - 1] to false
2: Set c[1], ..., c[218] to zero
3: for  $z = 0, \dots, 2^{18} - 1$  do
4:   if seen[ $z$ ] is false then
5:      $u \leftarrow z, \ell \leftarrow 0$ 
6:     repeat
7:       seen[ $u$ ]  $\leftarrow$  true,  $\ell \leftarrow \ell + 1$ 
8:        $u \leftarrow \text{next}(u)$ 
9:     until  $u = z$ 
10:     $c[\ell] \leftarrow c[\ell] + 1$ 
11:   end if
12: end for
13: return all  $(\ell, c[\ell])$  with  $c[\ell] \neq 0$ 
```

Correctness and cost. Proposition 9 supplies an inverse for T , so every state belongs to exactly one cycle. If z is unmarked, its cycle cannot contain a previously marked state: an earlier traversal would then have visited the entire same cycle, including z . The inner loop therefore marks exactly one new cycle and returns to z after its least period ℓ . Incrementing $c[\ell]$ counts that cycle once. The outer loop eventually visits every state, proving completeness. Each of the 2^{18} states is passed to next exactly once. The procedure uses 2^{18} Boolean entries for seen and 2^{18} integer entries for c .

Table 3. Complete cycle counts for T . Each pair of columns gives a length ℓ and the number c_ℓ of cycles of that length. Lengths not listed have count zero.

ℓ	c_ℓ	ℓ	c_ℓ	ℓ	c_ℓ
1	1	60	9	609	9
3	1	93	9	688	9
6	450	111	9	744	9
9	9	162	9	991	9
12	414	193	9	1369	9
14	3	249	9	1548	9
18	9	314	9	1636	18
24	135	377	9	1638	9
27	2	425	9	1936	9
30	9	435	9	2334	9
33	9	483	9	2479	9
42	15	569	9	2888	9
48	36	576	9	3009	9

Table 3 is the resulting histogram. In particular,

$$\sum_{\ell} c_{\ell} = 1336, \quad \sum_{\ell} \ell c_{\ell} = 262144 = 2^{18}, \quad \max\{\ell : c_{\ell} > 0\} = 3009.$$

The lengths dividing six are 1, 2, 3, 6, with counts 1, 0, 1, 450, respectively. Their cycles contain $1 + 3 + 6 \cdot 450 = 2704$ states, giving the count used in Section 8.

D QSH: Finite Certificates and Leaf Replacement

D.1 Exact classifications from finite certificates

Let $\mathcal{B}$ be a finite alphabet containing two distinct elements denoted by 0 and 1. The coordinate set is $\mathbb{Z}_4^3$, with the translation T from Section 7. For a coordinate partition $\mathcal{P}$, put

$$W_{\mathcal{P}}(\mathcal{B}) = \{X \in \mathcal{B}^{64} : X_i = X_j \text{ whenever } i, j \text{ lie in one block of } \mathcal{P}\}.$$

The block values are free and independent, so this set has $|\mathcal{B}|^{|\mathcal{P}|}$ elements. In the QSH instance, $\mathcal{B} = \mathcal{A}_w$ and we abbreviate this set to $W_{\mathcal{P}}$. Its binary dimension

Table 4. Exact partition conditions for the classification criterion. The integer word values 0 and 1 specify the QSH inputs completely.

State inspected	Number Induced partition
$E(\mathbf{1})$	1 $\mathcal{O}_1$
$E^{-1}(\mathbf{1})$	1 $\mathcal{O}_1$
$E(e_{a,0})$, all a	16 $\mathcal{D}$
$E(e_{a,\{0,1\}})$, all a	16 $\mathcal{D}$
$E(e_{a,\{0,2\}})$, all a	16 $\mathcal{O}_2$

is $w|\mathcal{P}|$. Constants and additional relations between block values are outside this class of spaces.

The partition induced by equality of coordinate values in a state Y groups coordinates exactly when their values in Y agree. Thus $Y \in W_{\mathcal{P}}(\mathcal{B})$ if and only if $\mathcal{P}$ refines that partition, where refinement means that every block of the first partition is contained in a block of the second. Write $\mathcal{O}_1, \mathcal{O}_2$ for the orbit partitions of T, T^2 , and $\mathcal{D}$ for the discrete partition. For $a = (a_0, a_1) \in \mathbb{Z}_4^2$ and $S \subseteq \mathbb{Z}_4$, let $e_{a,S}$ take value 1 at (a_0, a_1, z) with $z \in S$, and value 0 elsewhere. Put $e_{a,z} = e_{a,\{z\}}$, and let $\mathbf{1}$ be the constant state with word value 1.

Theorem 3 (Sufficient conditions for the exact classifications). *Suppose that a bijection $E : \mathcal{B}^{64} \rightarrow \mathcal{B}^{64}$ commutes with T and satisfies all exact partition conditions in Table 4. Then*

$$E(W_{\mathcal{P}}(\mathcal{B})) = W_{\mathcal{Q}}(\mathcal{B}) \iff \mathcal{P} = \mathcal{Q} \in \{\mathcal{O}_1, \mathcal{O}_2, \mathcal{D}\}.$$

If, in addition, the 16 multisets of word values in $E(e_{a,0})$ are pairwise different, then for pure coordinate permutations $P, Q \in S_{64}$,

$$E \circ P = Q \circ E \iff P = Q = T^k \text{ for some } k \in \mathbb{Z}_4.$$

Proof. Commutation gives $E(\text{Fix}(T^r)) \subseteq \text{Fix}(T^r)$ for $r = 1, 2$. Injectivity and finiteness turn these inclusions into equalities.

Suppose first that $E(W_{\mathcal{P}}(\mathcal{B})) = W_{\mathcal{Q}}(\mathcal{B})$. Cardinalities give $p = |\mathcal{P}| = |\mathcal{Q}| = q$. Every equality space defined above contains $\mathbf{1}$, so $E(\mathbf{1}) \in W_{\mathcal{Q}}(\mathcal{B})$ and $E^{-1}(\mathbf{1}) \in W_{\mathcal{P}}(\mathcal{B})$. The first two rows force both partitions to refine $\mathcal{O}_1$. Their blocks therefore lie inside the 16 four-point translation orbits.

The indicator of each $\mathcal{P}$ -block belongs to $W_{\mathcal{P}}(\mathcal{B})$. If a block is a singleton, its indicator is a translate of some $e_{a,0}$. Commutation and the third certificate row imply that its image has 64 different values. Hence $\mathcal{Q} = \mathcal{D}$, and $p = q$ also forces $\mathcal{P} = \mathcal{D}$.

In every other case each four-point orbit is partitioned without singletons, hence into one block of size four or two pairs. An adjacent pair has an indicator that is a translate of $e_{a,\{0,1\}}$, again forcing the discrete case. Only opposite pairs remain, so $p \leq 32$. If an opposite pair occurs, its indicator is a translate of $e_{a,\{0,2\}}$. The last row then forces $\mathcal{Q}$ to refine $\mathcal{O}_2$, giving $q \geq 32$. Thus $p = q = 32$, every

orbit in $\mathcal{P}$ splits into its opposite pairs, and $\mathcal{P} = \mathcal{Q} = \mathcal{O}_2$. If no pair occurs, $\mathcal{P} = \mathcal{O}_1$, and $p = q$ with refinement gives $\mathcal{Q} = \mathcal{O}_1$. These are all possibilities. The equalities for $\text{Fix}(T^r)$ and bijectivity establish the converse.

For the second assertion, let $e_0 = e_{(0,0),0}$ and write $P(e_0) = e_j$, the unit state at coordinate j . The relation $E(e_j) = Q(E(e_0))$ implies equality of their multisets of word values. By commutation, the four outputs $E(e_{a,z})$ with the same a have equal multisets of word values, whereas the added hypothesis separates the 16 translation orbits. Consequently $P(e_0) = T^k e_0$ for some $k \in \mathbb{Z}_4$. It follows that

$$Q(E(e_0)) = E(T^k e_0) = T^k(E(e_0)).$$

The 64 values of $E(e_0)$ are all different. Its stabilizer among coordinate permutations is the identity, so $Q = T^k$. Bijectivity and commutation now give $P = E^{-1}T^k E = T^k$. Conversely, commutation verifies all four pairs (T^k, T^k) .

D.2 Exact instantiation for the two prescribed cores

Lemma 2 (QSH certificate values). *For each $w \in \{32, 64\}$, the prescribed full core E_w satisfies the partition conditions of Table 4. The multisets of word values in the 16 outputs $E_w(e_{a,0})$ are pairwise different.*

Proof (Exact evaluation procedure). Represent every word by its integer in $[0, 2^w - 1]$, reduce each addition modulo 2^w , and implement the rotations and ordered groups in (20) and (19). Apply nine complete three-layer rounds followed by one column layer for $w = 32$, and eighteen complete rounds followed by one column layer for $w = 64$. This determines every computation below without any choice of samples.

Evaluate $E_w(\mathbf{1})$ and the 48 indicator inputs in the last three table rows, taking a in lexicographic order. Obtain $B = E_w^{-1}(\mathbf{1})$ by undoing the final column layer and then the complete rounds in reverse order. Within each ordered call to G , undo its eight assignments in reverse order: subtract to undo an addition, and replace $x \leftarrow \text{rotl}_w(x \oplus y, s)$ by $x \leftarrow \text{rotr}_w(x, s) \oplus y$. One further forward evaluation checks $E_w(B) = \mathbf{1}$. Bijectivity uniquely identifies the preimage.

For each state listed in the table, compare every pair of its 64 coordinates. The required equivalences are

$$\begin{aligned} i \sim_{\mathcal{O}_1} j &\iff x_0 = y_0 \text{ and } x_1 = y_1, \\ i \sim_{\mathcal{O}_2} j &\iff x_0 = y_0, \ x_1 = y_1, \ x_2 \equiv y_2 \pmod{2}, \\ i \sim_{\mathcal{D}} j &\iff i = j, \end{aligned}$$

where $i = 16x_0 + 4x_2 + x_1$ and $j = 16y_0 + 4y_2 + y_1$. The tests require equality exactly within the indicated blocks and inequality across blocks. Finally, sort each of the 16 arrays $E_w(e_{a,0})$ as a list of 64 integers, retaining multiplicities, and compare all $\binom{16}{2} = 120$ pairs of sorted lists. At each width, all specified partition conditions hold, and the 16 sorted lists are pairwise different. The procedure uses 50 evaluations of E_w and one evaluation of E_w^{-1} for the preimage row.

Proposition 6 and Lemma 2 give both classifications in Proposition 8 by Theorem 3. Injectivity also turns $E_w(W) \subseteq W$ into equality for a finite space W . Thus the invariant equality spaces $W_{\mathcal{P}}$ have precisely the dimensions $16w, 32w, 64w$. The proof concerns the complete map: it permits intermediate states to leave a candidate space and return to it. The classifications concern spaces defined solely by equalities between entire words and pure permutations of word positions, respectively.

D.3 Leaf replacement under the prescribed initialization

Write $m = 32w$ for the block length, $c = 16w$ for the complete CV width, and H^0 for the prescribed reset. QSH resets each leaf independently, uses the same flags for every complete non-root data chunk, and injects no chunk index into those calls [6, Secs. 4–5]. Suppose distinct 16-block strings C_0, C_1 , evaluated from H^0 with flags $(1, 0^{14}, 2)$, have the same complete CV V . For a permitted positive integer t , define

$$\mathcal{M}_t = \{C_{b_1} \parallel \cdots \parallel C_{b_t} : (b_1, \dots, b_t) \in \{0, 1\}^t\}. \quad (37)$$

Fixed chunk boundaries and $C_0 \neq C_1$ make all 2^t members distinct. Their common raw length $16tm$ requires two padding blocks, which form one final leaf. Let W_t be that leaf’s CV, evaluated from H^0 with flags $(1, 2)$. Every member has the same ordered leaf sequence $(V, \dots, V, W_t)$. Equal parent inputs, resets, tree partitions, and root flags then give equal complete digests. The argument uses equality of the entire transmitted CV; equality of the other 48 state words is unnecessary.

Proposition 11 (Shared digest computation). *Given C_0, C_1 and their complete CV V , the common digest of $\mathcal{M}_t$ can be computed using*

$$\lfloor \log_2 t \rfloor + \text{popcount}(t) + 2$$

additional compressions, where $\text{popcount}(t)$ is the number of set bits of t .

Proof. Put $d = \lfloor \log_2 t \rfloor$. Compute the non-root subtree CVs

$$A_0 = V, \quad A_{j+1} = \text{CV}(F_4(H^0, A_j \parallel A_j)), \quad 0 \leq j < d.$$

Here A_j represents 2^j consecutive copies of V , so the cache costs d compressions. Two further compressions give W_t . Initialize $Z = W_t$ and scan the set bits of t from least to most significant. For each set bit at position j , replace

$$Z \leftarrow \text{CV}(F_{f_j}(H^0, A_j \parallel Z)), \quad f_j = \begin{cases} 4, & j < d, \\ 12, & j = d. \end{cases}$$

Apply the prescribed digest truncation to the final Z .

To check the tree shape, write $t = 2^d + s$, where $0 \leq s < 2^d$. For $t + 1$ leaves, the root’s left subtree contains the largest power of two strictly below $t + 1$,

namely 2^d , and therefore has CV Λ_d . The right subtree contains s copies of V followed by W_t . Recursing on that suffix gives exactly the updates corresponding to the lower set bits of t . These updates compute non-root parents with flag 4, and only the final update has root flag 12. There is one update per set bit, proving the count.

To obtain the seed pair from the prescribed reset, fix any 15-block prefix P and compute $H_{15} = \Psi_P(H^0)$ with the legal flags $(1, 0^{14})$. The map

$$g_P(M) = \text{CV}(F_2(H_{15}, M)), \quad M \in \mathcal{A}_w^{32},$$

has $D = 2^{32w} = 2^{2c}$ inputs and at most $K = 2^{16w} = 2^c$ outputs. Sample $q = 2^{c/2} = 2^{8w}$ blocks independently and uniformly with replacement, and return only two distinct blocks with equal CVs. Lemma 1 gives a success probability greater than 0.39: its exponential term has argument greater than 499/1000, the term accounting for repeated sampled inputs is less than 1/1000, and (2) applies. The prefix is evaluated once; each sampled final block then requires one compression evaluation. At most q records, each holding a block and its CV, suffice.

The two chunks $C_i = P \parallel M_i$ give Corollary 4. With H^0 cached, its total compression budget is

$$15 + 2^{c/2} + \lfloor \log_2 t \rfloor + \text{popcount}(t) + 2. \quad (38)$$

For $t = 2$, the four messages $C_0C_0, C_0C_1, C_1C_0, C_1C_1$ have 32 raw blocks and a budget of $2^{c/2} + 19$ compressions. Their common digest needs two padding calls, one non-root parent call on $V \parallel V$, and one root call. Computing an uncached H^0 adds one evaluation of E_w .

Equation (38) counts a compact family description and one shared digest evaluation. Sampling, comparisons, and storage operations are additional costs, and explicitly writing all messages requires $t2^t$ data chunks. Requiring the padded length to fit the 64-bit interface gives $(16t + 2)m \leq 2^{64} - 1$, hence $t \leq 2^{50} - 1$ for $w = 32$ and $t \leq 2^{49} - 1$ for $w = 64$.

The map from a complete data chunk to its CV is identical at every position, so one pair of chunks with equal complete CVs can be reused throughout the tree. This is an application of CV replacement in a tree, whose validity depends on the node encoding and domain flags [1]. Joux's multicollision construction instead uses successive mergers of chaining states [9]. Equation (38) accounts for the single collision search and the shared tree computation. For fixed t it retains exponent $c/2$. In particular, the complete CV has 1024 bits in QSH-768 as well as QSH-1024: replacing it by the 768-bit final digest would invalidate leaf substitution. For an ordinary collision the generic CV route therefore does not lower the digest birthday exponent.

D.4 Explicit free-start witnesses

For each variant, take the following distinct raw byte strings, of lengths 96 and 160 bits:

$$\begin{aligned} M_0 &= 515348207265766965772041, \\ M_1 &= 51534820726576696577206d6573736167652042. \end{aligned}$$

Let $P_{i,0}, P_{i,1}$ be their two padded blocks, decoded as little-endian words. Their only chunk uses flags (1, 10), since END and ROOT apply to the second block. Define the complete reset

$$I^{(i)} = F_{1,P_{i,0}}^{-1} (F_{10,P_{i,1}}^{-1} (0^{64w})), \quad i = 0, 1. \quad (39)$$

This specifies both resets completely using the inverse in (23), without a search or an unspecified parameter. Forward evaluation cancels the two inverse maps, so the complete terminal state is zero and

$$\text{QSH}_{I^{(0)}}(M_0) = \text{QSH}_{I^{(1)}}(M_1) = 0^n.$$

The first reset words below certify that the two resets differ at each width. The same pair of resets for $w = 64$ serves the 768- and 1024-bit variants, which differ only in the final truncation after the reset is selected.

$n \ I_0^{(0)}$	$I_0^{(1)}$
512 fb0f95a4	b8b609bf
768, 1024 d863fe5d1b1d6233	f70cb15272fe6154

Table 5 gives all 64 words of each QSH-512 reset. Each row lists words $j, j+1, j+2, j+3$; these are integer word values, serialized little-endian when used as bytes. Together with the two messages and the common zero digest, the table is a complete numerical free-start collision witness.

Table 5. Complete QSH-512 resets for the free-start witness.

j	Words of $I^{(0)}$			Words of $I^{(1)}$			
0	fb0f95a4	3d2ff85b	1433f670	4747e436	b8b609bf	9fc75d3f	f713ba21
4	fddb7147	3c30c897	ba598769	fcf60333	3ba54abb	481880be	30cbd4ab
8	a0600d9d	c5cb49e1	f2a3317f	4bead8a4	9b5ffd8a	08671256	db6d83f7
12	b60ee66e	16e8c1f9	c496b149	6c825786	f4a006de	95e1ff10	5e5900d2
16	0a087fa3	a5487809	d601b63f	3c718dbc	76f90f8c	57d6af4f	bfad41e0
20	89e80224	5ba76f5a	6bf4dfc9	ded43af2	42d70bac	981931b6	2ca3341c
24	b457c278	4973b3c6	d681f2f0	14df0cb8	3c60479f	4f71d77d	26113071
28	3cfa117a	d7d067e5	69a7839a	03176a4b	b097482b	532c254d	0a62ea2f
32	1a756f63	54708317	19bf1aed	de50ce91	fd83e43f	fb8c2029	1c870f5e
36	fd6a223c	dc97113a	62d449d1	2fbdd9d7	b6b70283	852a16d5	1dbc4ca1
40	85691408	29ac773a	962d88cf	8bbb9552	56d1085b	2243182e	390b24fe
44	4cf42eb6	9cc6ea0b	ec001fe2	37219d80	ecaab4f5	672e079c	a06fc628
48	d1904054	c43aaefc	a3b687dc	db1e0889	87c79673	34da736c	d1d37b92
52	52196d46	4c66f138	4128d789	89dce0c5	2290f9d9	a42dde25	a6215617
56	93d44359	09f8bbc0	fe50092e	10745f21	e2a07330	2071afff	0f1e4442
60	61b7c30d	46cc41b7	23a43158	9576343a	5897f5bc	c1023073	63fb0bff
							f0f40820